

## **BACKGROUND OF THE INVENTION**

(1) **FIELD OF THE INVENTION:** The present invention relates to an electronic, self contained, automated bill payment machine and method of use incorporating real time or batch processing of the payment.

(2) **DESCRIPTION OF THE PRIOR ART:** In many instances, bills, such as utility bills, credit card payments, bills for subscriptions for newspapers, periodicals and the like are paid by mailing checks or by credit card payments and even electronic fund transfers from a bank or other financial institution authorized by the recipient of the bill or the "payor". One common method of paying utility bills is the presentment of a bill by a customer to a clerk or teller at a grocery store or other consumer environment. The bill is paid with cash, check or by means of a credit or debit card. The bill may contain scan line information or information in another format which identifies the customer's account number and gives data about the amounts owed, the due date, the nature of the utility, and the like. The teller or the clerk at the grocery or other store then scans the scan line information with a scanner which reads the information into a computer and the computer computes the amount due with a clerk manually inputting the amount paid. Alternatively, of course, these steps may be, and frequently are, accomplished by manual means without the use of the computer. The payor's account is updated with the payment information and the payment is credited to the account of the particular utility or other provider.

This system has many obvious disadvantages, one being that it is very labor-intensive and costly for the issuers, as well as the grocery and other stores which collect the payments. Another, equally onerous disadvantage is that the customers often wait in long lines to pay their

bills, and can pay them only when tellers or clerks are on duty.

The present invention addresses the problems of the prior art generally as above described.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

5           Fig. 1 is a schematic perspective illustration of the machine of the present invention, in kiosk format.

Figs. 2A and 2B together constitute a logic flow chart of the various method steps incorporated into the machine of the present invention.

### **SUMMARY OF THE INVENTION**

10           The present invention provides a machine for the payment in real time or batch of a bill. As used herein "bill" means a document in the hand of an operator which requires financial satisfaction. Typically, the bill will be issued and transmitted to a person owing the bill, hereinafter referred to as a "payor" by an entity referred to as a "billor". The payor may be an individual or a business, governmental or educational entity. The billor typically will be a  
15   utility company, but may be any other type of business entity, such as a retail store, outlet, service provider, such as a plumber, electrician, or the like.

The machine comprises a support structure which is preferably in the form of a body or base having a number of faces in the form of a "kiosk" structure. Electronic means within the support structure are provided for transmitting data in real time or batch such as through a  
20   dedicated telephone line, or the like to and from the machine pertaining to the payment of the bill. Means are secured to the support structure for electronically scanning the bill to identify and retrieve predeterminable data on the bill such as the name of the billor, the name of the

payor, the amount of the bill, the type of bill, the frequency of the bill, whether or not partial payment is accepted, and the like. A microprocessor is incorporated into a computer which includes a computer program for the operation of the computer to store and retrieve data pertaining to the payment of the bill and for generating commands to the machine for implementing the various steps in the operation of the machine and the practice of the method.

Means are provided in the structure for optically scanning a check issued by a financial institution, such as a bank, which is presented on behalf of the payor by the human operator of the machine for payment of the bill. Video monitor means, preferably a touch-sensitive screen, is provided for prompting and confirming various commands and steps in the operation of the machine and the method. Means are also provided for generating on the screen a video instruction image for operation of the screen by the operator, as, for example, the image of a human or the like. Means are also provided for generating an audio signal in a preselected language, i.e., English or other language, concurrently with the video instruction image generated on the screen, which may also be the touch screen or, alternatively, a separate screen.

A computer is programmed for computing data received through the optical reader and controlling the screen and the audio signal. The computer also applies, as programmed, the payment of the bill and electronically transmits data pertaining to the payment of the bill in real time or batch to the billor and, if a check is used for full or partial payment of the bill, to the financial institution to generate a real time or batch ACH transaction debit to the account of the payor.

In one embodiment of the invention, a method for payment by the operator for use of the computer system is provided in which a check may be used to initiate full or part payment

of the bill. The check is not required to be either signed or filled in, such as by filling in the amount of the check and the name of the payee along the line which typically will state "pay to the order of". The machine and method converts this to a self-service "ACH transaction". As used in the specification and in the claims, the phrase "ACH TRANSACTION" means the conversion of a conventional check drawn on a financial institution and converting it into an electronic check for payment through means of an automated banking clearing house (ACH). An electronic check is an electronic payment instruction which is digitally signed by the payor. Typically, a payor issues an electronic check with a digital signature to the payee. The payee endorses the check by providing a digital signature to the check which then passes the endorsed check electronically to the payee's bank. The payee's bank verifies the syntax of the check to make sure that it is correct, and also verifies the signature of the payee on the check. Thereafter, the payee bank converts the check into a format which is suitable for clearing with the Automatic Clearing House (ACH) of the Federal Reserve Net Settlement System. The payee bank issues a formatted ACH debit instruction containing the check to the ACH. At the designated time of day, the ACH performs the check settlements and eventually performs a funds transfer by sending a credit to the payee's bank and a debit to the payee institution. As contemplated herein, an ACH transaction converts an unsigned conventional check and the amount of the check not inserted on the check into a scanner and the data therefrom is processed and converted into an ACH transaction.

In actual commercial operation, the "payor" is a retail consumer who is not a professional computer operator. The machine and the method permit self-service bill payment by the operator/consumer. The service is provided 24 hours per day, 7 days per week, without

the assistance of an on-site trained operator. Assistance to the unskilled operator is provided by a call center professional who can remotely assist in completing the transaction.

The machine and method also permit the remote acceptance of cash by the computer from the payor. It provides money reconciliation for the retailer/host to secure the control and  
5 reconciliation over the cash accepted by the machine. It then provides for the ACH sweep of the cash depository account and automatic conversion of cash to electronic ACH to permit transfer of the funds.

An SQL data base located on the support structure runs independently of the central computer even if communications between the central computer and the machine are  
10 interrupted. The data base is periodically replicated to the central computer when communications are working and the data is automatically uploaded to the central machine for final processing.

### **DESCRIPTION OF THE PREFERRED EMBODIMENTS**

The computer of the machine includes a micro processor which may be any one of a  
15 number of commercially available integrated circuit micro processors, such as Compaq® (a registered trademark of Compaq Incorporated). It may be battery-powered or backed-up or may operate off of alternating current source and will have a random access memory ("RAM") and a read-only memory ("ROM") and is programmed as herein described.

Now with first reference to Fig. 1, there is shown a perspective illustration of the  
20 machine 100 of the present invention in kiosk format for practice of the method. As shown in Fig. 1, a base or lower kiosk housing 1 receives a companion upper kiosk housing 2 which, as shown, has a number of faces or off-sets 2a, 2b, and 2c. The faces 2b and 2c, as will be shown,

are duplicative with respect to receipt of operational components.

Each of the faces 2b, 2c contain a touch screen monitor of conventional form and readily available from a number of commercial sources. A "touch screen" is an input device which is used to acquire data for the computer through the computer program for the control of functions in the machine and method. A "touch" on a touch screen means that the touch screen senses the presence of an object, such as a tip of a finger of a human operator or another object, for example, a stylist, at and/or at a small distance from an active surface area ("field") of the touch screen. An output signal which, in general, is either an electrical, i.e. optical, signal is generated from the touch screen. The output signal may include information which is directly dependent on the position of the "touch" on the touch screen.

The active surface area on the screen may be arranged into predetermined regions and, when a particular region is "touched", the output signal may then depend on a unique identification code which refers to that particular region of the screen. An input component including the touch screen performs data processing on the output signal from the touch screen to provide a signal which is compatible with a predetermined format, as further described herein. Typical of such touch screen technology is as disclosed in U.S. Patent No. 4,550, 211 to Mabusth, entitled "Touch Sensitive Control Device."

Each of the faces 2b, 2c have in proximity to the touch screens 3 an electronic key pad 4 for input of various numerals required for identification of billing accounts, check numbers, and other manual insertion of data for use by the computer as hereinafter described.

An optical check reader 5 is provided on check face 2a and on a face (not shown) adjacent face 2c for introduction through a window 5a of a check to be optically scanned during

the payment process. Preferably, the optical scanner for reading of checks is as manufactured by RDM Corporation of Canada.

A telephone hand set 6 is also provided on face 2a in the event that the customer desires to communicate with a service assistant or operator for questions about the operation of the apparatus or method or to report errors or malfunctions.

A bill acceptor 7 also is provided on panel surface 2a for the insertion of paper currency to determine the genuineness of the currency as well its amount for purposes of calculating correct payment of the utility bill. The currency is moved over a path along which various optical, magnetic or edge sensing tests are performed. Such devices are commercially available from a number of sources, or well known to those skilled in the art and may be selected from a number of varying components. Typical of such prior art devices is that as generally disclosed in U.S. Patent No. 4,470,496 entitled "Control Circuit for Bill and Coin Changer," to Steiner. Such devices may be provided in a component which is programmed to dispense a cash voucher for the inserted paper currency when the amount of the paper currency and/or the check exceed the amount of the bill or bills to be paid.

A series of printers 8 are disposed on the face 1a of the housing 1 for printing of receipts and other information reflecting the payment transaction occurring through the apparatus.

As shown, the kiosk incorporating the present apparatus has at its top a television 9 which is controlled by a third computer within the support structure. This computer may be operated with a number of operating software systems, such as Windows 2000 and MSInternet Explorer. The computer also contains an SQL database which keeps track of the advertisements that are being shown via digital video on the television 9. The advertisers have the ability to

connect to central host computer systems via the web which contains a database of demographic profiles for the neighborhoods where each of the machines are located. The advertiser is able to select the demographic criteria which it desires, such as income, race, language, housing value, sex, geographic region and many other categories which the advertiser desires to target with the advertisement. The advertiser is then permitted to rank and select the machines with locations most like the preferred or targeted demographic profile requirements. The advertiser then binds an electronic contract for the particular machines on which it wishes to advertise and selects the time slots and number of times it wishes to run the advertisement on the television. The advertiser then produces the advertisement to the technical specification and downloads the advertisement to a central computer system joining a plurality of the self-service machines. The advertisement is then electronically "pushed" out to the individual, selective machines.

The software preferably utilized to implement the present invention may be any one of a machine code, such as visual basic, or Microsoft ASP. The logic and sub-routines utilized to form the machinery and method disclosed herein is set forth below:

```
-- MainSecurity replication job
```

```
*****
```

```
begin transaction
```

```
DECLARE @JobID BINARY(16)
```

```
DECLARE @ReturnCode INT
```

```
SELECT @ReturnCode = 0
```

```
-- if the standard merge job category doesn't exist, create it
```

```
if (select count(*) from msdb.dbo.syscategories where name = N'REPL-Merge') < 1  
    execute msdb.dbo.sp_add_category N'REPL-Merge'
```

```
-- if this job doesn't exist, create it
```



```

select @JobID = job_id from msdb.dbo.sysjobs where (name =
N'ASKSQL01-MainSecurity-MainSecurity-KL00001')
if (@JobID is NULL)
BEGIN
5   execute @ReturnCode = msdb.dbo.sp_add_job @job_id = @JobID OUTPUT, @job_name
   = N'ASKSQL01-MainSecurity-MainSecurity-KL00001', @enabled = 1, @start_step_id = 1,
   @notify_level_eventlog = 2, @notify_level_email = 0, @notify_level_netsend = 0,
   @notify_level_page = 0, @delete_level = 0, @description = N'MainSecurity database
10  replication for KL00001', @category_name = N'REPL-Merge', @owner_login_name = N'sa'
   if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

   -- step one, 'start agent' message
   execute @ReturnCode = msdb.dbo.sp_add_jobstep @job_id = @JobID , @step_id = 1,
   @cmdexec_success_code = 0, @on_success_action = 3, @on_success_step_id = 0,
15  @on_fail_action = 3, @on_fail_step_id = 0, @retry_attempts = 0, @retry_interval = 0,
   @os_run_priority = 0, @flags = 0, @step_name = N'Log Reader Agent startup message.',
   @subsystem = N'TSQL', @command = N'sp_MSadd_merge_history @perfmon_increment =
   0, @agent_id = 155, @runstatus = 1,
   @comments = "Starting agent.", @server =
20  N'ASKSQL01', @database_name = N'distribution'
   if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

   -- step two, run agent (standard two-way replication)
   execute @ReturnCode = msdb.dbo.sp_add_jobstep @job_id = @JobID , @step_id = 2,
25  @cmdexec_success_code = 0, @on_success_action = 1, @on_success_step_id = 0,
   @on_fail_action = 3, @on_fail_step_id = 0, @retry_attempts = 10, @retry_interval = 1,
   @os_run_priority = 0, @flags = 0, @step_name = N'Run agent.', @subsystem = N'Merge',
   @command = N'-Publisher [ASKSQL01] -PublisherDB [MainSecurity] -Publication
   [MainSecurity] -Subscriber [KL00001] -SubscriberDB [LocalSecurity] -Distributor
30  [ASKSQL01] -DistributorSecurityMode 1 ', @server = N'ASKSQL01', @database_name =
   N'distribution'
   if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

   -- step three, look for non-logged shutdown
35  execute @ReturnCode = msdb.dbo.sp_add_jobstep @job_id = @JobID , @step_id = 3,
   @cmdexec_success_code = 0, @on_success_action = 2, @on_success_step_id = 0,
   @on_fail_action = 2, @on_fail_step_id = 0, @retry_attempts = 0, @retry_interval = 0,
   @os_run_priority = 0, @flags = 0, @step_name = N'Detect nonlogged agent shutdown.',
   @subsystem = N'TSQL', @command = N'sp_MSdetect_nonlogged_shutdown @subsystem =
40  "Merge", @agent_id = 155', @server = N'ASKSQL01', @database_name = N'distribution'
   if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

   -- start job at step one

```

```

execute @ReturnCode = msdb.dbo.sp_update_job @job_id = @JobID, @start_step_id = 1
if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

```

```

-- schedule job to run nightly

```

```

5   execute @ReturnCode = msdb.dbo.sp_add_jobschedule @job_id = @JobID, @name =
N'Replication agent schedule.', @enabled = 1, @freq_type = 4, @freq_interval = 1,
@freq_subday_type = 1, @freq_subday_interval = 1, @freq_relative_interval = 2,
@freq_recurrence_factor=0, @active_start_date=20010326, @active_end_date=99991231,
@active_start_time = 0, @active_end_time = 235959
10  if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

```

```

-- job runs on ASKSQL01

```

```

execute @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @JobID, @server_name =
N'asksql01'
15  if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

```

```

END

```

```

20  commit transaction
    goto EndSave
QuitWithRollback:
    if (@@TRANCOUNT > 0) rollback transaction
EndSave:
25  GO

```

-- MainKiosk replication job

-  
\*\*\*\*\*

5 begin transaction

DECLARE @JobID BINARY(16)

DECLARE @ReturnCode INT

SELECT @ReturnCode = 0

10

-- if the standard merge job category doesn't exist, create it

if (select count(\*) from msdb.dbo.syscategories where name = N'REPL-Merge') < 1

execute msdb.dbo.sp\_add\_category N'REPL-Merge'

15

-- if this replication merge job doesn't exist, create it

select @JobID = job\_id from msdb.dbo.sysjobs where (name = N'ASKSQL01-MainKiosk-MainKiosk-KL00001')

if (@JobID is NULL)

BEGIN

20

execute @ReturnCode = msdb.dbo.sp\_add\_job @job\_id = @JobID OUTPUT, @job\_name = N'ASKSQL01-MainKiosk-MainKiosk-KL00001', @enabled = 1, @start\_step\_id = 1, @notify\_level\_eventlog = 2, @notify\_level\_email = 0, @notify\_level\_netsend = 0, @notify\_level\_page = 0, @delete\_level = 0, @description = N'MainKiosk database replication for KL0001', @category\_name = N'REPL-Merge', @owner\_login\_name = N'sa'

25

if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

-- step one, 'start agent' message

execute @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id = @JobID, @step\_id = 1, @cmdexec\_success\_code = 0, @on\_success\_action = 3, @on\_success\_step\_id = 0, @on\_fail\_action = 3, @on\_fail\_step\_id = 0, @retry\_attempts = 0, @retry\_interval = 0, @os\_run\_priority = 0, @flags = 0, @step\_name = N'Log Reader Agent startup message.', @subsystem = N'TSQL', @command = N'sp\_MSadd\_merge\_history @perfmon\_increment = 0, @agent\_id = 154, @runstatus = 1,

30

@comments = "Starting agent.", @server =

35

N'ASKSQL01', @database\_name = N'distribution'

if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

-- step two, run the agent with ExchangeType 1 for one-way inbound replication only

execute @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id = @JobID, @step\_id = 2, @cmdexec\_success\_code = 0, @on\_success\_action = 1, @on\_success\_step\_id = 0, @on\_fail\_action = 3, @on\_fail\_step\_id = 0, @retry\_attempts = 10, @retry\_interval = 1, @os\_run\_priority = 0, @flags = 0, @step\_name = N'Run agent.', @subsystem = N'Merge', @command = N'-Publisher [ASKSQL01] -PublisherDB [MainKiosk] -Publication [MainKiosk]

40

```
-Subscriber [KL00001] -SubscriberDB [LocalKiosk] -Distributor [ASKSQL01]
-DistributorSecurityMode 1 -ExchangeType 1', @server = N'ASKSQL01'
if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback
```

```
5  -- step three, look for non-logged shutdown
    execute @ReturnCode = msdb.dbo.sp_add_jobstep @job_id = @JobID , @step_id = 3,
    @cmdexec_success_code = 0, @on_success_action = 2, @on_success_step_id = 0,
    @on_fail_action = 2, @on_fail_step_id = 0, @retry_attempts = 0, @retry_interval = 0,
    @os_run_priority = 0, @flags = 0, @step_name = N'Detect nonlogged agent shutdown.',
10  @subsystem = N'TSQL', @command = N'sp_MSdetect_nonlogged_shutdown @subsystem =
    "Merge", @agent_id = 154', @server = N'ASKSQL01', @database_name = N'distribution'
    if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback
```

```
-- start job at step one
15  execute @ReturnCode = msdb.dbo.sp_update_job @job_id = @JobID, @start_step_id = 1
    if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback
```

```
-- schedule job to run half-hourly
    execute @ReturnCode = msdb.dbo.sp_add_jobschedule @job_id = @JobID, @name =
20  N'Replication agent schedule.', @enabled = 1, @freq_type = 4, @freq_interval = 1,
    @freq_subday_type = 4, @freq_subday_interval = 30, @freq_relative_interval = 2,
    @freq_recurrence_factor = 0, @active_start_date = 20010326, @active_end_date = 99991231,
    @active_start_time = 0, @active_end_time = 235959
    if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback
```

```
25  -- job runs on ASKSQL01
    execute @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @JobID, @server_name =
    N'asksql01'
    if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback
30
```

END

```
commit transaction
35  goto EndSave
QuitWithRollback:
    if (@@TRANCOUNT > 0) rollback transaction
EndSave:
```

```
40  GO
```

-- Replication failure e-mail job

-

\*\*\*\*\*

5 begin transaction

DECLARE @JobID BINARY(16)

DECLARE @ReturnCode INT

SELECT @ReturnCode = 0

10

-- if job category doesn't exist, create it

if (select count(\*) from msdb.dbo.syscategories where name = N'REPL-Alert Response') < 1  
execute msdb.dbo.sp\_add\_category N'REPL-Alert Response'

15

-- if this job doesn't exist, create it

select @JobID = job\_id from msdb.dbo.sysjobs where (name = N'Replication Failure e-mail')  
if (@JobID is NULL)

BEGIN

20

execute @ReturnCode = msdb.dbo.sp\_add\_job @job\_id = @JobID OUTPUT, @job\_name  
= N'Replication Failure e-mail', @enabled = 1, @start\_step\_id = 1, @notify\_level\_eventlog =  
2, @notify\_level\_email = 0, @notify\_level\_netsend = 0, @notify\_level\_page = 0,  
@delete\_level = 0, @description = N'Send e-mail using SMTP when replication agent fails.',  
@category\_name = N'REPL-Alert Response', @owner\_login\_name = N'sa'  
if (@@ERROR > 0 OR @ReturnCode > 0) goto QuitWithRollback

25

execute @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id = @JobID, @step\_id = 1,  
@cmdexec\_success\_code = 0, @on\_success\_action = 1, @on\_success\_step\_id = 0,  
@on\_fail\_action = 2, @on\_fail\_step\_id = 0, @retry\_attempts = 0, @retry\_interval = 1,  
@os\_run\_priority = 0, @flags = 0, @step\_name = N'Send E-mail', @subsystem = N'TSQL',  
30 @command = N'if (datepart(hh, getdate()) between 7 and 20)

begin

declare @KioskName varchar(100)

select top 1

35

@KioskName = substring(name, 30, 7) + "@askharris.com"

from

distribution.dbo.MSmerge\_agents

join distribution.dbo.MSmerge\_history on

distribution.dbo.MSmerge\_history.agent\_id = distribution.dbo.MSmerge\_agents.id

40

where

name like "%MainKiosk%" and

runstatus = 6

group by name

```

order by max(time) desc

exec pr_sendmail @KioskName,"techs@askharris.com","", "Replication Failure","This
is an automated alert."
5      if (datepart(hh, getdate()) > 14)
            exec pr_sendmail @KioskName,"7137208607@archwireless.net","", "Replication
Failure","They got the message too..."
end
', @database_name = N'MainKiosk'
10      if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

execute @ReturnCode = msdb.dbo.sp_update_job @job_id = @JobID, @start_step_id = 1
if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

15      execute @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @JobID, @server_name =
N'asksql01'
            if (@@ERROR <> 0 OR @ReturnCode <> 0) goto QuitWithRollback

END
20      commit transaction
            goto EndSave
QuitWithRollback:
            if (@@TRANCOUNT > 0) rollback transaction
25      EndSave:

GO

```

```
-- TWC PMT summary report
-- cash and check subtotals
```

```
-
*****
```

5

```
-- RL 02/04/2001 : made from CHW payment report
-- RL 03/29/2001 : add datetime values to join conditions
```

```
use MainKiosk
```

10

```
go
```

```
drop proc pr_TWC_PMT_report
go
```

15

```
create proc pr_TWC_PMT_report
as begin
```

```
-- overhead for file creation
```

20

```
create table ##TWC_PMT
(id int identity, OutputLine varchar(200))
delete from ##TWC_PMT
declare @FileName varchar(100)
```

25

```
select @FileName = 'c:\askHarris_TWC_PMThuman_' + replace(convert(varchar, getdate(),
106), ' ', ',') + '-' + right('0' + cast(datepart(hh, getdate()) as varchar(2)), 2) + right('0' +
cast(datepart(mi, getdate()) as varchar(2)), 2) + '.txt'
```

```
declare @OutputLine varchar(200)
```

30

```
declare @QueryCount int
declare @QueryTotal money
declare @SummaryCount int
declare @SummaryTotal money
```

35

```
select @SummaryCount = 0
select @SummaryTotal = 0
```

40

```
insert into ##TWC_PMT (OutputLine)
values ('')
insert into ##TWC_PMT (OutputLine)
values ('askHarris.com')
insert into ##TWC_PMT (OutputLine)
```

```

values ('PAYMENT SUBTOTALS')
insert into ##TWC_PMT (OutputLine)
values ('for Time Warner')
insert into ##TWC_PMT (OutputLine)
5 values ('Generated ' + convert(varchar, getdate()))
insert into ##TWC_PMT (OutputLine)
values ('')

```

```

10 declare Something cursor for
select
    cast((PaymentType + ' Transactions') as char(20))
    + ' ' +
    cast(isnull(count(*),0) as char(10))
15 + ' ' +
    convert(char(15), isnull(sum(PaymentAmount),0),0),
    isnull(count(*),0), sum(PaymentAmount)
from
(select
20     case count(*)
        when 1 then max(pt.PaymentTypeName)
        else 'Mixed'
        end as PaymentType,
        (sum(prt.PaymentAmount) - max(bdt.ServiceAmt) - IsNull(max(bdt.Donation), 0)) as
25 PaymentAmount
from
    BillDetailTransaction bdt
    JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
    bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
30     JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId
    and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
    JOIN PaymentType pt on pt.PaymentTypeID = prt.PaymentTypeid
    JOIN ReceiptBill rb on rb.BillPayID = bp.BillPayID and rb.KioskPCID =
    bp.KioskPCID and rb.DateStamp = bp.DateStamp
35     JOIN ReceiptDetailTransaction rdt on rdt.ReceiptDetailID = rb.ReceiptDetailID and
    rb.KioskPCID = rdt.KioskPCID and rb.DateStamp = rdt.DateTimeofReceiptIssue
where
    bdt.CompanyAccountTypeID = 10 and
    bdt.QBTed = 10
40 group by bdt.billdetailid, bdt.kioskpcid) as foo
group by PaymentType

open Something

```



```

fetch from Something into @OutputLine, @QueryCount, @QueryTotal

while @@fetch_status = 0
begin
5      select @SummaryCount = @SummaryCount + @QueryCount
      select @SummaryTotal = @SummaryTotal + @QueryTotal

      insert into ##TWC_PMT (OutputLine)
      values (@OutputLine)
10     fetch next from Something into @OutputLine, @QueryCount, @QueryTotal
end

close Something
deallocate Something

15  insert into ##TWC_PMT (OutputLine)
      values (")
      insert into ##TWC_PMT (OutputLine)
      values ('Total Transactions ' + cast(@SummaryCount as char(10)) + ' ' + convert(char(15),
20  @SummaryTotal))

      -- print contents of temp table to disk file
      declare @BCPCommand varchar(200)
25  select @BCPCommand = 'bcp "SELECT OutputLine FROM ##TWC_PMT order by id"
      queryout "' + @FileName + "' /T /c'
      EXEC master..xp_cmdshell @BCPCommand

      select @BCPCommand = 'move ' + @FileName + '
30  \\ASKVTS01\Transmissions\outgoing\TWC\'
      EXEC master..xp_cmdshell @BCPCommand

      insert into
35  MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
      values
      (getdate(), 0, 'TWC PMT report', 'File generated successfully.')

40  drop table ##TWC_PMT

      end -- create proc

```

-- SWB NSF report

\*\*\*\*\*

-- RL 01/20/2001 : start

5 -- RL 01/23/2001 : add total to end of report

-- RL 01/25/2001 : send output to text file on ASKVTS01

-- RL 01/27/2001 : use QBTEd value to choose records to process

-- RL 01/29/2001 : convert to stored procedure

-- RL 03/29/2001 : add datetime values to join conditions

10

use MainKiosk

go

drop proc pr\_SWB\_NSF

15 go

create proc pr\_SWB\_NSF

as begin

20 -- overhead for file creation

create table ##SWB\_NSF

(id int identity, OutputLine varchar(200))

delete from ##SWB\_NSF

declare @FileName varchar(100)

25

select @FileName = 'c:\askHarris\_SWB\_NSF\_' + replace(convert(varchar, getdate(), 106), ' ', '-') + '-' + right('0' + cast(datepart(hh, getdate()) as varchar(2)), 2) + right('0' + cast(datepart(mi, getdate()) as varchar(2)), 2) + '.txt'

30 declare @OutputLine varchar(200)

declare @DetailAmount money

declare @TotalAmount money

35 insert into ##SWB\_NSF (OutputLine)  
values ('')

insert into ##SWB\_NSF (OutputLine)  
values ('askHarris.com')

insert into ##SWB\_NSF (OutputLine)

40 values ('RETURNED CHECK REPORT')

insert into ##SWB\_NSF (OutputLine)

values ('for Southwestern Bell')

insert into ##SWB\_NSF (OutputLine)

```

values ('Generated ' + convert(varchar, getdate()))
insert into ##SWB_NSF (OutputLine)
values (")
insert into ##SWB_NSF (OutputLine)
5 values (' Payment ')
insert into ##SWB_NSF (OutputLine)
values (' Date Account # Amount ')
insert into ##SWB_NSF (OutputLine)
values ('-----')
10

declare NSFCursor cursor for
select
'' +
15 convert(char(10), prt.DateTimeofPayment, 101)
+ ' ' +
bdt.BillAccountNumber
+ ' ' +
convert(char(12), prt.PaymentAmount - bdt.ServiceAmt, 0),
20 prt.PaymentAmount - bdt.ServiceAmt
from
PaymentRecordTransaction prt
join ReturnedChecks rc on prt.PaymentRecordID = rc.PaymentRecordID and
prt.KioskPCID = rc.KioskPCID
25 join BillPay bp on prt.PaymentRecordID = bp.PaymentRecordID and prt.KioskPCID
= bp.KioskPCID and prt.DateTimeofPayment = bp.DateStamp
join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and bdt.KioskPCID
= bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
where
30 bdt.companyaccounttypeid = 3 and
rc.QBTed = 10
order by
prt.DateTimeofPayment

35 select @TotalAmount = 0
open NSFCursor
fetch from NSFCursor into @OutputLine, @DetailAmount

while @@fetch_status = 0
40 begin
select @TotalAmount = @TotalAmount + @DetailAmount
insert into ##SWB_NSF (OutputLine)
values (@OutputLine)

```

```

        fetch next from NSFCursor into @OutputLine, @DetailAmount
    end

    close NSFCursor
5   deallocate NSFCursor

    insert into ##SWB_NSF (OutputLine)
    values (")
    insert into ##SWB_NSF (OutputLine)
10  values ('=====')
    insert into ##SWB_NSF (OutputLine)
    values ('    TOTAL AMOUNT:    ' + convert(char(12), @TotalAmount, 0))

    -- print contents of temp table to disk file
15  declare @BCPCommand varchar(200)
    select @BCPCommand = 'bcp "SELECT OutputLine FROM ##SWB_NSF order by id"
    queryout "' + @FileName + "' /T /c'
    EXEC master..xp_cmdshell @BCPCommand

20  select @BCPCommand = 'move ' + @FileName + '
    \\ASKVTS01\Transmissions\outgoing\SWB\'
    EXEC master..xp_cmdshell @BCPCommand

    insert into
25      MainSecurity.dbo.JobRC
        (DateTime, ReturnCode, JobTxt, TxtComment)
    values
        (getdate(), 0, 'SWB NSF file', 'File generated successfully.')

30  drop table ##SWB_NSF

    end -- create proc

```

-- Time Warner Cable PMT file generation

\*\*\*\*\*

```
5  -- RL 11/29/2000 : file generated with no example given
   -- RL 01/05/2001 : switch from unions to cursor/print, format changes, batch by system, output
   -- RL 01/09/2001 : added live $$ADD stuff
   -- RL 01/25/2001 : changed output file to go to ASKVTS01
10  -- RL 01/27/2001 : use QBTed value to choose records to process
   -- RL 01/29/2001 : convert to stored procedure
   -- RL 02/03/2001 : correct for mixed payments (one payment, one service charge)
   -- RL 02/10/2001 : prevent reporting of number <=0 ($1 payment, $1 svc chg)
   -- RL 03/29/2001 : add datetime values to join conditions
15
   use MainKiosk
   go

   drop proc pr_TWC_PMT
20  go

   create proc pr_TWC_PMT
   as begin

25  -- temporary table overhead
   create table ##twc_pmt
   (id int identity, line char(40))

   -- batch by system number
30  declare @SystemNbr char(4)

   -- file record count includes headers/trailers
   declare @FileRecordCount int
   declare @OutputPart1 varchar(200)
35  declare @OutputPart2 varchar(200)
   declare @FileOutputLine varchar(200)
   select @FileRecordCount = 0

   -- batch totals
40  declare @BatchDollarAmount float
   declare @BatchRecordCount int
   declare @DetailDollarAmount float
```

```

-- variables for wire transfer at end of procedure
declare @FileTotal float
select @FileTotal = 0
declare @WireOffset int
5  if datename(dw,getdate()) in ('Thursday','Friday')
    select @WireOffset = 4
    else
        select @WireOffset = 2

10  -- begin with CSG specified $$ADD statement
    insert into ##twc_pmt (line) values
        ($$ADD ID=CAD1AHL BID="S8111AHL"      )
    insert into ##twc_pmt (line) values
        (          )

15  declare @SystemNbrFetchStatus int

    declare SystemNbrCursor cursor for
    select distinct
20      left(BillAccountNumber,4) as SystemNbr
    from
        BillDetailTransaction
    where
        CompanyAccountTypeID = 10 and
25      QBTed = 10

    open SystemNbrCursor
    fetch from SystemNbrCursor into @SystemNbr

30  select @SystemNbrFetchStatus = @@fetch_status

    while @SystemNbrFetchStatus = 0
    begin

35      select @FileRecordCount = @FileRecordCount + 1
        select @BatchRecordCount = 0
        select @BatchDollarAmount = 0

        -- record type 1
        -- Batch Header
40      insert into ##twc_pmt (line) values
            (      '1'
                + --      as RecordTypeCode,

```

```

        @SystemNbr
        + --    as CableSystemNumber,
        right('0' + cast(datepart(mm,getDate()) as varchar(2)),2)+
        right('0' + cast(datepart(dd,getDate()) as varchar(2)),2)+
5         cast(right(datepart(yy,getDate()),2) as char(2))
        + --    as DateOfTape,
        ' '

        + --    as CablePrinNumber,
        ' '

10         + --    as CableAgentNumber,
        ' '

        -- as Filler
    )

15     -- record type 5
    -- Detail
    declare DetailCursor cursor for
    select
        '5'
20         + --    as RecordTypeCode,
        max(bdt.BillAccountNumber)
        + --    as SubscriberAccountNumber,
        right('0' + cast(datepart(mm,max(prt.DateTimeofPayment)) as varchar(2)),2)+
        right('0' + cast(datepart(dd,max(prt.DateTimeofPayment)) as varchar(2)),2)+
25         cast(right(datepart(yy,max(prt.DateTimeofPayment)),2) as char(2))
        + --    as TransactionDate,
        right('0000000' + cast(cast(round(((sum(prt.PaymentAmount) -
max(bdt.ServiceAmt)) * 100),0) as int) as varchar(7)),7)
        + --    as TransactionAmount,
30         'P'
        --    as TransactionCode,
        as OutputPart1,
        -- unique id (x8) goes here
        --    as TransactionIdentifier,
35         ''
        as OutputPart2,
        (sum(prt.PaymentAmount) - max(bdt.ServiceAmt))
        as DetailDollarAmount

    from
40         BillDetailTransaction bdt,
        PaymentRecordTransaction prt,
        BillPay bp

    where

```





```

        ,
        + --      as CablePrinNumber,
        ,
        + --      as CableAgentNumber,
5         ,
        --      as Filler
    )

    select @FileTotal = @FileTotal + @BatchDollarAmount

10    close DetailCursor
    deallocate DetailCursor

    -- don't forget to fetch the next one
15    fetch next from SystemNbrCursor into @SystemNbr
    select @SystemNbrFetchStatus = @@fetch_status
end

    close SystemNbrCursor
20    deallocate SystemNbrCursor

    if (@FileRecordCount % 2) = 1
        insert into ##twc_pmt (line) values (

25    -- write wire transfer number to table
    insert into
        MainSecurity.dbo.WireTransfers
        (Payee, WireDate, WireAmount, Comment)
    values
30    ('Time Warner', dateadd(day,@WireOffset,getdate()), @FileTotal, 'Generated ' +
        convert(varchar,getdate(),101))

    -- print temp table contents out to file
    declare @FileName varchar(100)
35    declare @BCPCommand varchar(200)

    select @FileName = 'c:\askHarris_TWC_PMT_' + replace(convert(varchar, getdate(), 106),
        ' ', '-') + '-' + right('0' + cast(datepart(hh,getdate()) as varchar(2)),2) + right('0' +
        cast(datepart(mi,getdate()) as varchar(2)),2) + '.rpt'
40    select @BCPCommand = 'bcp "SELECT a.line + b.line FROM ##twc_pmt a, ##twc_pmt b
        where b.id = a.id + 1 and (a.id % 2) = 1" queryout "' + @FileName + '" /T /c'
        EXEC master..xp_cmdshell @BCPCommand

```

```
select @BCPCommand = 'move ' + @FileName + '  
\\ASKVTS01\Transmissions\outgoing\TWC\  
EXEC master..xp_cmdshell @BCPCommand
```

```
5  insert into  
    MainSecurity.dbo.JobRC  
    (DateTime, ReturnCode, JobTxt, TxtComment)  
    values  
    (getdate(), 0, 'TWC PMT file', 'File generated successfully.')  
10 drop table ##twc_pmt  
  
end -- create proc
```

```

-- procedure to send mail by SMTP
-
*****
-- RL 02/27/2001 : file created from sp_OA* example and Bert's email.vbs
5  -- RL 02/27/2001 : convert to stored procedure
-- *****
-- example use: pr_sendmail @MailFrom, @MailTo, @MailCc, @MailSubject, @MailBody
- -   e x a m p l e   u s e :   p r _ s e n d m a i l
'sql@askharris.com','rlynch@servcomp.com','techs@askharris.com',
10  -- 'Test Subject','This is a test message.'
-- *****

use MainKiosk
go

15  drop procedure pr_sendmail
go

create procedure pr_sendmail
20      @MailFrom varchar(100) = 'SQLnotify@servcomp.com',
      @MailTo varchar(100) = "",
      @MailCc varchar(100) = "",
      @MailSubject varchar(100) = "",
      @MailBody varchar(1000) = "

25  as begin

      if @MailTo = ""
          goto BailOut

30  -- integer pointers for objects
      declare @objMessage int
      -- declare @objConfiguration int

      declare @ErrorValue int
35  declare @src varchar(255), @desc varchar(255)

      -- create objMessage
      exec @ErrorValue = sp_OAcreate 'CDO.Message', @objMessage out
40  if @ErrorValue <> 0
      begin
          print 'objMessage error'
          exec sp_OAGetErrorInfo @objMessage, @src out, @desc out

```

```

select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
end

/*
5  -- create objConfiguration
exec @ErrorValue = sp_OAcreate 'CDO.Configuration', @objConfiguration out
if @ErrorValue <> 0
    begin
        print 'objConfiguration error'
10    exec sp_OAGetErrorInfo @objConfiguration, @src out, @desc out
        select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
        end

    exec @ErrorValue = sp_OASetProperty @objConfiguration,
15    'Fields("http://schemas.microsoft.com/cdo/configuration/smtpserver")', '172.16.1.7'
    if @ErrorValue <> 0
        begin
            print 'cdoSMTPServer error'
            exec sp_OAGetErrorInfo @objConfiguration, @src out, @desc out
20        select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
            end

        -- set colFields values
        exec @ErrorValue = sp_OASetProperty @objConfiguration,
25    'Fields("http://schemas.microsoft.com/cdo/configuration/smtpserverport")', '25'
        if @ErrorValue <> 0
            begin
                print 'cdoSMTPServerPort error'
                exec sp_OAGetErrorInfo @objConfiguration, @src out, @desc out
30        select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
            end

        -- set colFields values
        exec @ErrorValue = sp_OASetProperty @objConfiguration,
35    'Fields("http://schemas.microsoft.com/cdo/configuration/sendusing")', '2'
        if @ErrorValue <> 0
            begin
                print 'cdoSendUsingMethod error'
                exec sp_OAGetErrorInfo @objConfiguration, @src out, @desc out
40        select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
            end

        -- set colFields values

```

```

exec @ErrorValue = sp_OASetProperty @objConfiguration,
'Fields("http://schemas.microsoft.com/cdo/configuration/smtpconnectiontimeout")', '30'
if @ErrorValue <> 0
begin
5   print 'cdoSMTPConnectionTimeout error'
   exec sp_OAGetErrorInfo @objConfiguration, @src out, @desc out
   select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
end

10  -- Update colFields
exec @ErrorValue = sp_OAMethod @objConfiguration, 'Fields.Update'
if @ErrorValue <> 0
begin
   print 'colFields.Update error'
15  exec sp_OAGetErrorInfo @objConfiguration, @src out, @desc out
   select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
end

   */

20  -- set objMessage values
exec @ErrorValue = sp_OASetProperty @objMessage, 'From', @MailFrom
if @ErrorValue <> 0
begin
   print 'objMessage.From error'
25  exec sp_OAGetErrorInfo @objMessage, @src out, @desc out
   select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
end

   -- set objMessage values
30  exec @ErrorValue = sp_OASetProperty @objMessage, 'To', @MailTo
if @ErrorValue <> 0
begin
   print 'objMessage.To error'
   exec sp_OAGetErrorInfo @objMessage, @src out, @desc out
35  select Error=convert(varbinary(4),@ErrorValue),Source=@src,Description=@desc
end

   -- set objMessage values
exec @ErrorValue = sp_OASetProperty @objMessage, 'Cc', @MailCc
40  if @ErrorValue <> 0
begin
   print 'objMessage.Cc error'
   exec sp_OAGetErrorInfo @objMessage, @src out, @desc out

```





-- Memo File Generator routine  
-- depending on day of week

\*\*\*\*\*

5 -- RL 01/30/2001 : broke into separate companies, added day-of-week check  
-- RL 02/04/2001 : copied from kiosk file generator

use MainKiosk  
go

10 -- drop proc pr\_memo\_file\_gen

create proc pr\_memo\_file\_gen  
as  
15 begin

declare @WeekDay varchar(10)

20 select @WeekDay = datename(dw,getdate())  
-- do not run this process on Saturday or Sunday morning  
if @WeekDay in ('Sunday', 'Saturday')  
begin  
insert into

25 MainSecurity.dbo.JobRC  
(DateTime, ReturnCode, JobTxt, TxtComment)  
values  
(getdate(), -1, 'Memo Generator', 'ABORTED: Memo Generator should not run on ' +  
@WeekDay + '.')  
30 goto finish  
end

insert into MainSecurity.dbo.JobRC  
(DateTime, ReturnCode, JobTxt, TxtComment)  
35 values  
(getdate(), 0, 'Memo Generator', '\*\* Memo generation starting. \*\*')

-- HL&P bills  
40 insert into MainSecurity.dbo.JobRC  
(DateTime, ReturnCode, JobTxt, TxtComment)  
values  
(getdate(), 0, 'Memo Generator', 'Start HL&P Memo generation.')



```

exec pr_HLP_MEMO

insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
5  values
    (getdate(), 0, 'Memo Generator', 'End HL&P Memo generation.')

-- CHW bills
insert into MainSecurity.dbo.JobRC
10  (DateTime, ReturnCode, JobTxt, TxtComment)
    values
    (getdate(), 0, 'Memo Generator', 'Start CHW memo generation.')

exec pr_CHW_MEMO
15
insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
    values
20  (getdate(), 0, 'Memo Generator', 'End CHW memo generation.')

insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
    values
25  (getdate(), 0, 'Memo Generator', '** Memo generation completed. **')

finish:

return
30
end -- create proc

```

```
-- Memo Cutoff routine
-- depending on day of week
```

```
-
*****
```

```
5  -- RL 01/30/2001 : broke into separate companies, added day-of-week check
   -- RL 02/03/2001 : made from kiosk cutoff file
   -- RL 03/19/2001 : eliminate partial transactions
   -- RL 03/29/2001 : add datetime values to join conditions
```

```
10  use MainKiosk
    go
```

```
    drop proc pr_memo_cutoff
    go
```

```
15  create proc pr_memo_cutoff
    as
    begin
```

```
20  declare @WeekDay varchar(10)
```

```
    select @WeekDay = datename(dw,getdate())
```

```
    -- do not run this process if any records are in "to be processed/in process" status
```

```
25  declare @TenCount int
```

```
    select @TenCount = isnull(count(*),0) from paymentrecordtransaction where qbted = 10
```

```
    select @TenCount = @TenCount + isnull(count(*),0) from billdetailtransaction where qbted = 10
```

```
30  select @TenCount = @TenCount + isnull(count(*),0) from cashvoucherissuedetailtransaction
    where qbted = 10
```

```
    select @TenCount = @TenCount + isnull(count(*),0) from returnedchecks where qbted = 10
```

```
    if @TenCount <> 0
```

```
35  begin
    insert into
```

```
        MainSecurity.dbo.JobRC
        (DateTime, ReturnCode, JobTxt, TxtComment)
    values
```

```
40  (getdate(), -1, 'Memo Cutoff', 'ABORTED: Kiosk Cleanup has not been run.')
    goto finish
end
```

```

-- do not run this process on Saturday or Sunday morning
if @WeekDay in ('Saturday', 'Sunday')
begin
insert into
5      MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
      values
      (getdate(), -1, 'Memo Cutoff', 'ABORTED: Cutoff should not run on ' + @WeekDay +
      ':')
10     goto finish
end

-- eliminate zero-amount payments
      update PaymentRecordTransaction
15     set QBTed = 90
      where (QBTed is null or QBTed = 0 or QBTed = 15) and PaymentAmount = 0

-- eliminate zero-company bills
      update BillDetailTransaction
20     set QBTed = 90
      where (QBTed is null or QBTed = 0 or QBTed = 15) and CompanyAccountTypeID =
      0

-- eliminate partial transactions based on payments
25     update PaymentRecordTransaction
      set QBTed = 90
      where rowguid in
      (select prt.rowguid
      from
30          paymentrecordtransaction prt
      left join billpay bp on prt.paymentrecordid = bp.paymentrecordid and
      prt.kioskpcid = bp.kioskpcid and prt.DateTimeofPayment = bp.DateStamp
      left join billdetailtransaction bdt on bdt.billdetailid = bp.billdetailid and
      bdt.kioskpcid = bp.kioskpcid and bdt.DateTimeofBillScan = bp.DateStamp
35     left join receiptbill rb on bp.billpayid = rb.billpayid and bp.kioskpcid =
      rb.kioskpcid and bp.DateStamp = rb.DateStamp
      left join receiptdetailtransaction rdt on rb.receiptdetailid =
      rdt.receiptdetailid and rb.kioskpcid = rdt.kioskpcid and rb.DateStamp =
      rdt.DateTimeofReceiptIssue
40     where
      (bp.billpayid is null or
      bdt.billdetailid is null or
      rb.receiptbillid is null or

```

rdt.receiptdetailid is null) and  
prt.qbted is null)

-- eliminate partial transactions based on bills

```

5      update BillDetailTransaction
      set QBTed = 90
      where rowguid in
          (select bdt.rowguid
           from
10              billdetailtransaction bdt
              left join billpay bp on bdt.billdetailid = bp.billdetailid and bdt.kioskpcid
              = bp.kioskpcid and bdt.DateTimeofBillScan = bp.DateStamp
              left join paymentrecordtransaction prt on prt.paymentrecordid =
              bp.paymentrecordid and prt.kioskpcid = bp.kioskpcid and prt.DateTimeofPayment =
15              bp.DateStamp
              left join receiptbill rb on bp.billpayid = rb.billpayid and bp.kioskpcid =
              rb.kioskpcid and bp.DateStamp = rb.DateStamp
              left join receiptdetailtransaction rdt on rb.receiptdetailid =
              rdt.receiptdetailid and rb.kioskpcid = rdt.kioskpcid and rb.DateStamp =
20              rdt.DateTimeofReceiptIssue
              where
                  (bp.billpayid is null or
                   bdt.billdetailid is null or
                   rb.receiptbillid is null or
                   rdt.receiptdetailid is null) and
25                  prt.qbted is null)

```

-- HL&P bills

```

30      update BillDetailTransaction
      set QBTed = 10
      where (QBTed is null or QBTed = 0) and CompanyAccountTypeID = 1
      insert into MainSecurity.dbo.JobRC
          (DateTime, ReturnCode, JobTxt, TxtComment)
      values
35      (getdate(), @@rowcount, 'Memo Cutoff', 'HL&P bill cutoff completed.')

```

-- CHW bills

```

      update BillDetailTransaction
      set QBTed = 10
40      where (QBTed is null or QBTed = 0) and CompanyAccountTypeID = 4
      insert into MainSecurity.dbo.JobRC
          (DateTime, ReturnCode, JobTxt, TxtComment)
      values

```

```
(getdate(), @@rowcount, 'Memo Cutoff', 'CHW bill cutoff completed.')
```

```
insert into MainSecurity.dbo.JobRC  
  (DateTime, ReturnCode, JobTxt, TxtComment)
```

```
5  values  
    (getdate(), 0, 'Memo Cutoff', '** All bill cutoffs completed. **')
```

```
finish:
```

```
10 return
```

```
end -- create proc
```

Printed on 04/23/2014

```
-- Memo Cleanup routine
-- depending on day of week
-
*****
```

```
5 -- RL 01/30/2001 : broke into separate companies, added day-of-week check
-- RL 02/03/2001 : made from memo cutoff file
```

```
use MainKiosk
go
```

```
10 -- drop proc pr_memo_cleanup
```

```
create proc pr_memo_cleanup
as
15 begin
```

```
declare @WeekDay varchar(10)
```

```
20 select @WeekDay = datename(dw,getdate())
```

```
-- do not run this process on Saturday nights
if @WeekDay in ('Saturday', 'Sunday')
begin
insert into
```

```
25 MainSecurity.dbo.JobRC
(DateTime, ReturnCode, JobTxt, TxtComment)
values
(getdate(), -1, 'Memo Cutoff', 'ABORTED: Cutoff should not run on' + @WeekDay +
'.')
30 goto finish
end
```

```
-- HL&P bills
35 update BillDetailTransaction
set QBTed = 15
where (QBTed = 10) and CompanyAccountTypeID = 1
insert into MainSecurity.dbo.JobRC
(DateTime, ReturnCode, JobTxt, TxtComment)
40 values
(getdate(), @@rowcount, 'Memo Cutoff', 'HL&P bill cutoff completed.')
```

```
-- CHW bills
```

```

update BillDetailTransaction
set QBTed = 15
where (QBTed = 10) and CompanyAccountTypeID = 4
insert into MainSecurity.dbo.JobRC
5      (DateTime, ReturnCode, JobTxt, TxtComment)
values
      (getdate(), @@rowcount, 'Memo Cutoff', 'CHW bill cutoff completed.')

10  insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
values
      (getdate(), 0, 'Memo Cutoff', '** All bill cutoffs completed. **')

15  finish:

      return

      end -- create proc
20

```

```

-- Kiosk next-day routine
-- sets QBTed to 30
-- *****
-- RL 02/11/2001 : file created

```

5

```

use MainKiosk
go

```

```

-- drop proc pr_kiosk_nextday

```

10

```

create proc pr_kiosk_nextday
as
begin

```

15

```

    update billdetailtransaction
    set QBTed = 30
    where QBTed = 20

```

20

```

    update paymentrecordtransaction
    set QBTed = 30
    where QBTed = 20

```

25

```

    update cashvoucherissuedetailtransaction
    set QBTed = 30
    where QBTed = 20

```

30

```

    update returnedchecks
    set QBTed = 30
    where QBTed = 20

end -- create proc

```



```

-- Kiosk File Generator routine
-- depending on day of week
-
*****
5
-- RL 01/30/2001 : broke into separate companies, added day-of-week check

use MainKiosk
go
10
-- drop proc pr_kiosk_file_gen

create proc pr_kiosk_file_gen
as
15
begin

declare @WeekDay varchar(10)

select @WeekDay = datename(dw,getdate())
20
-- do not run this process on Saturday nights
if @WeekDay = 'Saturday'
begin
insert into
25
    MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
    values
    (getdate(), -1, 'File Generator', 'ABORTED: File Generator should not run on Saturday.')
goto finish
30
end

insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
values
35
    (getdate(), 0, 'File Generator', '** File generation starting. **')

-- HL&P bills
insert into MainSecurity.dbo.JobRC
40
    (DateTime, ReturnCode, JobTxt, TxtComment)
    values
    (getdate(), 0, 'File Generator', 'Start HL&P file generation.')

```

```

print 'HLP'
-- PMT
    exec pr_HLP_PMT
    exec pr_HLP_PMT_report
5  -- NSF
    --      exec pr_HLP_NSF

    insert into MainSecurity.dbo.JobRC
10      (DateTime, ReturnCode, JobTxt, TxtComment)
    values
        (getdate(), 0, 'File Generator', 'End HL&P file generation.')

    -- Entex bills
15  -- do not process on Friday nights
    if @WeekDay <> 'Friday'
    begin
        insert into MainSecurity.dbo.JobRC
            (DateTime, ReturnCode, JobTxt, TxtComment)
20      values
            (getdate(), 0, 'File Generator', 'Start Entex file generation.')

    print 'Entex'
    -- PMT
25      exec pr_Entex_PMT
        exec pr_Entex_PMT_report
    -- NSF
    --      exec pr_Entex_NSF

30      insert into MainSecurity.dbo.JobRC
            (DateTime, ReturnCode, JobTxt, TxtComment)
        values
            (getdate(), 0, 'File Generator', 'End Entex file generation.')

    end
35
    -- CHW bills
    -- do not process on Sunday nights
    if @WeekDay <> 'Sunday'
    begin
40      insert into MainSecurity.dbo.JobRC
            (DateTime, ReturnCode, JobTxt, TxtComment)
        values
            (getdate(), 0, 'File Generator', 'Start CHW file generation.')

```

```

print 'CHW'
-- PMT
-- NSF
5      exec pr_CHW_PMT_NSF
      exec pr_CHW_PMT_report
--      exec pr_CHW_NSF_report

      insert into MainSecurity.dbo.JobRC
          (DateTime, ReturnCode, JobTxt, TxtComment)
10      values
          (getdate(), 0, 'File Generator', 'End CHW file generation.')
      end

15      -- TWC bills
      -- do not process on Sunday nights
      if @WeekDay <> 'Sunday'
      begin
          insert into MainSecurity.dbo.JobRC
20              (DateTime, ReturnCode, JobTxt, TxtComment)
          values
              (getdate(), 0, 'File Generator', 'Start TWC file generation.')

      print 'TWC'
25      -- PMT
          exec pr_TWC_PMT
          exec pr_TWC_PMT_report
      -- NSF
      --      exec pr_TWC_NSF
30
          insert into MainSecurity.dbo.JobRC
              (DateTime, ReturnCode, JobTxt, TxtComment)
          values
              (getdate(), 0, 'File Generator', 'End TWC file generation.')
35      end

      -- Chase ACH file
      -- (including SWB bills)
      -- do not process on Sunday nights
40      if @WeekDay <> 'Sunday'
      begin
          insert into MainSecurity.dbo.JobRC
              (DateTime, ReturnCode, JobTxt, TxtComment)

```

```

        values
            (getdate(), 0, 'File Generator', 'Start Chase/SWB file generation.')

    print 'Chase'
5    -- PMT
        exec pr_Chase_ACH
    -- NSF
    --     exec pr_SWB_NSF

10    insert into MainSecurity.dbo.JobRC
        (DateTime, ReturnCode, JobTxt, TxtComment)
        values
            (getdate(), 0, 'File Generator', 'End Chase/SWB file generation.')

    end

15    insert into MainSecurity.dbo.JobRC
        (DateTime, ReturnCode, JobTxt, TxtComment)
        values
            (getdate(), 0, 'File Generator', '** File generation completed. **')

20    finish:

    return

25    end -- create proc

```

```

-- Kiosk Cutoff routine
-- depending on day of week
-
*****
5  -- RL 01/30/2001 : broke into separate companies, added day-of-week check
   -- RL 03/02/2001 : code to ignore zero-company bills
   -- RL 03/19/2001 : code to remove partial transactions
   -- RL 03/29/2001 : add datetime values to join conditions

10  use MainKiosk
    go

    drop proc pr_kiosk_cutoff
    go

15  create proc pr_kiosk_cutoff
    as
    begin

20  declare @WeekDay varchar(10)

    select @WeekDay = datename(dw,getdate())

    -- do not run this process on Saturday nights
25  if @WeekDay = 'Saturday'
    begin
    insert into
        MainSecurity.dbo.JobRC
        (DateTime, ReturnCode, JobTxt, TxtComment)
30  values
        (getdate(), -1, 'Kiosk Cutoff', 'ABORTED: Cutoff should not run on Saturday.')
    goto finish
    end

35  -- eliminate zero-amount payments
        update PaymentRecordTransaction
        set QBTed = 90
        where (QBTed is null or QBTed = 0 or QBTed = 15) and PaymentAmount = 0

40  -- eliminate zero-company bills
        update BillDetailTransaction
        set QBTed = 90
        where (QBTed is null or QBTed = 0 or QBTed = 15) and CompanyAccountTypeID =

```

0

-- eliminate partial transactions based on payments

```
5      update PaymentRecordTransaction
      set QBTed = 90
      where rowguid in
          (select prt.rowguid
           from
               paymentrecordtransaction prt
10          left join billpay bp on prt.paymentrecordid = bp.paymentrecordid and
prt.kioskpcid = bp.kioskpcid and prt.DateTimeofPayment = bp.DateStamp
          left join billdetailtransaction bdt on bdt.billdetailid = bp.billdetailid and
bdt.kioskpcid = bp.kioskpcid and bdt.DateTimeofBillScan = bp.DateStamp
          left join receiptbill rb on bp.billpayid = rb.billpayid and bp.kioskpcid =
15 rb.kioskpcid and bp.DateStamp = rb.DateStamp
          left join receiptdetailtransaction rdt on rb.receiptdetailid =
rdt.receiptdetailid and rb.kioskpcid = rdt.kioskpcid and rb.DateStamp =
rdt.DateTimeofReceiptIssue
          where
20          (bp.billpayid is null or
bdt.billdetailid is null or
rb.receiptbillid is null or
rdt.receiptdetailid is null) and
prt.qbtcd is null)
```

-- eliminate partial transactions based on bills

```
25      update BillDetailTransaction
      set QBTed = 90
      where rowguid in
30      (select bdt.rowguid
       from
           billdetailtransaction bdt
           left join billpay bp on bdt.billdetailid = bp.billdetailid and bdt.kioskpcid
= bp.kioskpcid and bdt.DateTimeofBillScan = bp.DateStamp
35      left join paymentrecordtransaction prt on prt.paymentrecordid =
bp.paymentrecordid and prt.kioskpcid = bp.kioskpcid and prt.DateTimeofPayment =
bp.DateStamp
           left join receiptbill rb on bp.billpayid = rb.billpayid and bp.kioskpcid =
rb.kioskpcid and bp.DateStamp = rb.DateStamp
40      left join receiptdetailtransaction rdt on rb.receiptdetailid =
rdt.receiptdetailid and rb.kioskpcid = rdt.kioskpcid and rb.DateStamp =
rdt.DateTimeofReceiptIssue
           where
```

```

5      (bp.billpayid is null or
      bdt.billdetailid is null or
      rb.receiptbillid is null or
      rdt.receiptdetailid is null) and
      prt.qbtcd is null)

```

```

-- HL&P bills
      update BillDetailTransaction
10      set QBTcd = 10
      where (QBTcd is null or QBTcd = 0 or QBTcd = 15) and CompanyAccountTypeID =
      1
      insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
15      values
      (getdate(), @@rowcount, 'Kiosk Cutoff', 'HL&P bill cutoff completed.')

```

```

-- Entex bills
-- do not process on Friday nights
20 if @WeekDay <> 'Friday'
begin
      update BillDetailTransaction
      set QBTcd = 10
      where (QBTcd is null or QBTcd = 0 or QBTcd = 15) and CompanyAccountTypeID in
25      (2,9)
      insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
      values
      (getdate(), @@rowcount, 'Kiosk Cutoff', 'Entex bill cutoff completed.')
30 end

```

```

-- CHW bills
-- do not process on Sunday nights
if @WeekDay <> 'Sunday'
35 begin
      update BillDetailTransaction
      set QBTcd = 10
      where (QBTcd is null or QBTcd = 0 or QBTcd = 15) and CompanyAccountTypeID =
      4
40      insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
      values
      (getdate(), @@rowcount, 'Kiosk Cutoff', 'CHW bill cutoff completed.')

```

end

-- TWC bills

5 -- do not process on Sunday nights

if @WeekDay <> 'Sunday'

begin

update BillDetailTransaction

set QBTed = 10

10 where (QBTed is null or QBTed = 0 or QBTed = 15) and CompanyAccountTypeID =

10

insert into MainSecurity.dbo.JobRC

(DateTime, ReturnCode, JobTxt, TxtComment)

values

15 (getdate(), @@rowcount, 'Kiosk Cutoff', 'TWC bill cutoff completed.')

end

-- SWB bills

-- do not process on Sunday nights

20 if @WeekDay <> 'Sunday'

begin

update BillDetailTransaction

set QBTed = 10

where (QBTed is null or QBTed = 0 or QBTed = 15) and CompanyAccountTypeID =

25 3

insert into MainSecurity.dbo.JobRC

(DateTime, ReturnCode, JobTxt, TxtComment)

values

(getdate(), @@rowcount, 'Kiosk Cutoff', 'SWB bill cutoff completed.')

30 end

-- Payments (for Chase ACH)

-- do not process on Sunday nights

if @WeekDay <> 'Sunday'

35 begin

update PaymentRecordTransaction

set QBTed = 10

where (QBTed is null or QBTed = 0)

insert into MainSecurity.dbo.JobRC

40 (DateTime, ReturnCode, JobTxt, TxtComment)

values

(getdate(), @@rowcount, 'Kiosk Cutoff', 'Chase payment cutoff completed.')



```

update CashVoucherIssueDetailTransaction
set QBTed = 10
where (QBTed is null or QBTed = 0)
insert into MainSecurity.dbo.JobRC
5      (DateTime, ReturnCode, JobTxt, TxtComment)
values
      (getdate(), @@rowcount, 'Kiosk Cutoff', 'Cash voucher cutoff completed.')

update ReturnedChecks
10    set QBTed = 10
      where (QBTed is null or QBTed = 0)
insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
values
15    (getdate(), @@rowcount, 'Kiosk Cutoff', 'Chase NSF cutoff completed.')
end

insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
20    values
      (getdate(), 0, 'Kiosk Cutoff', '** All bill cutoffs completed. **')

finish:
25    return

end -- create proc

```

```

-- Kiosk Cleanup routine
-- depending on day of week
-
*****
5  -- RL 01/30/2001 : broke into separate companies, added day-of-week check
   -- RL 03/09/2001 : removed day-of-week check

use MainKiosk
go
10  -- drop proc pr_kiosk_cleanup

create proc pr_kiosk_cleanup
as
15  begin

insert into
    MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
20  values
    (getdate(), 0, 'Kiosk Cleanup', '** Kiosk Cleanup started. **')

-- HL&P bills
update BillDetailTransaction
25  set QBTed = 20
    where (QBTed = 10) and CompanyAccountTypeID = 1
insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
values
30  (getdate(), @@rowcount, 'Kiosk Cleanup', 'HL&P bill cleanup completed.')

-- HL&P NSF checks
update ReturnedChecks
set QBTed = 20
35  where (QBTed = 10) and rowguid in
    (select rc.rowguid
     from
         ReturnedChecks rc
         join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
40  rc.KioskPCID = bp.KioskPCID
         join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
bp.KioskPCID = bdt.KioskPCID
     where

```

```

        bdt.CompanyAccountTypeID = 1)
insert into MainSecurity.dbo.JobRC
        (DateTime, ReturnCode, JobTxt, TxtComment)
values
5         (getdate(), @@rowcount, 'Kiosk Cleanup', 'HL&P NSF cleanup completed.')

-- Entex bills
-- do not process on Friday nights
10         update BillDetailTransaction
           set QBTed = 20
           where (QBTed = 10) and CompanyAccountTypeID in (2,9)
insert into MainSecurity.dbo.JobRC
           (DateTime, ReturnCode, JobTxt, TxtComment)
15         values
           (getdate(), @@rowcount, 'Kiosk Cleanup', 'Entex bill cleanup completed.')

-- Entex NSF checks
           update ReturnedChecks
20         set QBTed = 20
           where (QBTed = 10) and rowguid in
           (select rc.rowguid
            from
                ReturnedChecks rc
25             join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
rc.KioskPCID = bp.KioskPCID
                join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
bp.KioskPCID = bdt.KioskPCID
           where
30             bdt.CompanyAccountTypeID in (2,9))
insert into MainSecurity.dbo.JobRC
           (DateTime, ReturnCode, JobTxt, TxtComment)
           values
35         (getdate(), @@rowcount, 'Kiosk Cleanup', 'Entex NSF cleanup completed.')

-- CHW bills
-- do not process on Sunday nights
           update BillDetailTransaction
40         set QBTed = 20
           where (QBTed = 10) and CompanyAccountTypeID = 4
insert into MainSecurity.dbo.JobRC
           (DateTime, ReturnCode, JobTxt, TxtComment)

```

```

values
    (getdate(), @@rowcount, 'Kiosk Cleanup', 'CHW bill cleanup completed.')

-- CHW NSF checks
5      update ReturnedChecks
      set QBTed = 20
      where (QBTed = 10) and rowguid in
      (select rc.rowguid
      from
10         ReturnedChecks rc
      join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
rc.KioskPCID = bp.KioskPCID
      join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
bp.KioskPCID = bdt.KioskPCID
15      where
      bdt.CompanyAccountTypeID = 4)
      insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
      values
20      (getdate(), @@rowcount, 'Kiosk Cleanup', 'CHW NSF cleanup completed.')

-- TWC bills
-- do not process on Sunday nights
25      update BillDetailTransaction
      set QBTed = 20
      where (QBTed = 10) and CompanyAccountTypeID = 10
      insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
30      values
      (getdate(), @@rowcount, 'Kiosk Cleanup', 'TWC bill cleanup completed.')

-- TWC NSF checks
      update ReturnedChecks
35      set QBTed = 20
      where (QBTed = 10) and rowguid in
      (select rc.rowguid
      from
      ReturnedChecks rc
40      join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
rc.KioskPCID = bp.KioskPCID
      join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
bp.KioskPCID = bdt.KioskPCID

```

```

where
    bdt.CompanyAccountTypeID = 10)

insert into MainSecurity.dbo.JobRC
5      (DateTime, ReturnCode, JobTxt, TxtComment)
values
    (getdate(), @@rowcount, 'Kiosk Cleanup', 'TWC NSF cleanup completed.')

-- SWB bills
10  -- do not process on Sunday nights
    update CompanyAccountTypeMaster
    set Notes = cast((cast(Notes as int) + 1) as varchar)
    where CompanyAccountTypeID = 3

15  insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
values
    (getdate(), @@rowcount, 'Kiosk Cleanup', 'SWB EDI control number updated.')

20  update BillDetailTransaction
    set QBTed = 20
    where (QBTed = 10) and CompanyAccountTypeID = 3
    insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
25  values
    (getdate(), @@rowcount, 'Kiosk Cleanup', 'SWB bill cleanup completed.')

    update ReturnedChecks
    set QBTed = 20
30  where (QBTed = 10) and rowguid in
    (select rc.rowguid
    from
        ReturnedChecks rc
        join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
35  rc.KioskPCID = bp.KioskPCID
        join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
        bp.KioskPCID = bdt.KioskPCID
    where
        bdt.CompanyAccountTypeID = 3)
40  insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
values
    (getdate(), @@rowcount, 'Kiosk Cleanup', 'SWB NSF cleanup completed.')

```

```

-- Chase payments
-- do not process on Sunday nights
    update PaymentRecordTransaction
    set QBTed = 20
5      where (QBTed = 10)
    insert into MainSecurity.dbo.JobRC
        (DateTime, ReturnCode, JobTxt, TxtComment)
    values
        (getdate(), @@rowcount, 'Kiosk Cleanup', 'Chase payment cleanup completed.')
10
    update CashVoucherIssueDetailTransaction
    set QBTed = 20
    where (QBTed = 10)
    insert into MainSecurity.dbo.JobRC
15      (DateTime, ReturnCode, JobTxt, TxtComment)
    values
        (getdate(), @@rowcount, 'Kiosk Cleanup', 'Cash voucher cleanup completed.')

    insert into MainSecurity.dbo.JobRC
20      (DateTime, ReturnCode, JobTxt, TxtComment)
    values
        (getdate(), 0, 'Kiosk Cleanup', '** All bill cleanups completed. **')

    finish:
25      return

    end -- create proc
30

```

```
-- HL&P PMT report
-- Human readable report with receipt info
-
```

```
*****
```

5

```
-- RL 01/10/2001 : created query for example report
-- RL 01/20/2001 : revised to report format
-- RL 01/25/2001 : send output to text file on ASKVTS01
-- RL 01/27/2001 : use QBTed value to choose records to process
```

10

```
-- RL 01/29/2001 : convert to stored procedure
-- RL 03/03/2001 : catch up with changes to PMT file and display total at bottom of report
-- RL 03/29/2001 : add datetime values to join conditions
```

```
use MainKiosk
```

15

```
go
```

```
drop proc pr_HLP_PMT_report
go
```

20

```
create proc pr_HLP_PMT_report
as begin
```

```
-- overhead for file creation
```

```
create table ##HLP_PMT
```

25

```
(id int identity, OutputLine varchar(200))
```

```
delete from ##HLP_PMT
```

```
declare @FileName varchar(100)
```

30

```
select @FileName = 'c:\askHarris_HLP_PMT\human_' + replace(convert(varchar, getdate(),
106), ' ', ',') + '-' + right('0' + cast(datepart(hh, getdate()) as varchar(2)), 2) + right('0' +
cast(datepart(mi, getdate()) as varchar(2)), 2) + '.txt'
```

```
declare @OutputLine varchar(200)
```

35

```
declare @FileTotal money
```

```
declare @PayID int
```

```
declare @KioskPCID int
```

```
select @FileTotal = 0
```

40

```
insert into ##HLP_PMT (OutputLine)
```

```
values ('')
```

```
insert into ##HLP_PMT (OutputLine)
```

```

values ('askHarris.com')
insert into ##HLP_PMT (OutputLine)
values ('PAYMENT REPORT')
insert into ##HLP_PMT (OutputLine)
5 values ('for Reliant HL&P')
insert into ##HLP_PMT (OutputLine)
values ('Generated ' + convert(varchar, getdate()))
insert into ##HLP_PMT (OutputLine)
values ('')
10 insert into ##HLP_PMT (OutputLine)
values (' Date      Account #      Amount  Name                      TDL #   Store   Term
Receipt')
insert into ##HLP_PMT (OutputLine)
values
15 ('-----')

-- get list of multi-pay payment records, with totals for each multi-pay
-- treat single-pay as a type of multipay, with MultiPaySetCount of 1
20 declare @MultiPayFetchStatus int
declare @MultiPayBillID int
declare @ReportedPayment float

declare MultiPaySet cursor for
25 select
      prt.PaymentRecordID as PayID,
      prt.KioskPCID as KioskPCID
from
      BillDetailTransaction bdt
30 join BillPay bp on bp.KioskPCID = bdt.KioskPCID and bp.BillDetailID =
      bdt.BillDetailID and bp.DateStamp = bdt.DateTimeofBillScan
      join PaymentRecordTransaction prt on bp.KioskPCID = prt.KioskPCID and
      prt.PaymentRecordID = bp.PaymentRecordid and prt.DateTimeofPayment = bp.DateStamp
      JOIN MainSecurity.dbo.KioskPCInfo kpc on bdt.KioskPCID = kpc.KioskPCID
35 JOIN MainSecurity.dbo.KioskInfoMaster kim on kim.KioskInfoID = kpc.KioskID
      JOIN MainSecurity.dbo.StoreChainProfileMaster scp on kim.StoreChainProfileID =
      scp.StoreChainProfileID
      JOIN MainSecurity.dbo.StoreMaster sm on kim.StoreID = sm.StoreID
where
40 bdt.CompanyAccountTypeID = 1 and
      bdt.QBTed = 10
group by
      prt.paymentrecordid, prt.KioskPCID

```



```

order by
    prt.KioskPCID,
    count(distinct bp.billdetailid) desc

```

```

5  -- loop over payment records
   -- this is the batch loop
   open MultiPaySet
   fetch from MultiPaySet
   into @PayID, @KioskPCID
10  select @MultiPayFetchStatus = @@fetch_status

   while @MultiPayFetchStatus = 0
   begin

15      -- for each payment record, get bill records
      -- and data to report to PMT file
      declare @PaymentRecordFetchStatus int
      declare PaymentRecordSet cursor for
      select

20          convert(char(10), bdt.DateTimeofBillScan, 101)
          + ' ' +
          left(bdt.BillAccountNumber,13)
          + ' ' +
          case bdt.BillDetailID
25              when @MultiPayBillID then convert(char(12), isnull(bdt.AptPayAmt,
prt.PaymentAmount),0)
              else convert(char(12), (isnull(bdt.AptPayAmt, prt.PaymentAmount) -
bdt.ServiceAmt),0)
              end
          + ' ' +
30          cast(replace(isnull(cit.Name,' [Cash Payment] '), '$', ' ') as char(30))
          + ' ' +
          left(isnull(cit.IdentificationNumber, ' '),8)
          + ' ' +
35          cast(upper(left(scp.StoreChainName,4)) as char(4)) + ' ' +
          cast(right(sm.StoreName,3) as char(3))
          + ' ' +
          right('0000' + cast(prt.KioskPCID as varchar(4)), 4)
          + ' ' +
40          rdt.ReceiptNumber,
          bdt.BillDetailID as MultiPayBillID,
          case bdt.BillDetailID
              when @MultiPayBillID then isnull(bdt.AptPayAmt, prt.PaymentAmount)

```

```

else isnull(bdt.AptPayAmt, prt.PaymentAmount) - bdt.ServiceAmt
end as ReportedPayment
from
    PaymentRecordTransaction prt
5   join BillPay bp on prt.PaymentRecordID = bp.PaymentRecordID and
    prt.KioskPCID = bp.KioskPCID and prt.DateTimeofPayment = bp.DateStamp
    join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and
    bdt.KioskPCID = bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
    join ReceiptBill rb on rb.KioskPCID = bp.KioskPCID and rb.BillPayId =
10  bp.BillPayID and rb.DateStamp = bp.DateStamp
    join ReceiptDetailTransaction rdt on rdt.KioskPCID = rb.KioskPCID and
    rdt.ReceiptDetailID = rb.ReceiptDetailID and rdt.DateTimeofReceiptIssue = rb.DateStamp
    join MainSecurity.dbo.KioskPCInfo kpc on kpc.KioskPCID = prt.KioskPCID
    join MainSecurity.dbo.KioskInfoMaster kim on kpc.KioskID = kim.KioskInfoID
15  join MainSecurity.dbo.StoreChainProfileMaster scp on kim.StoreChainProfileID
    = scp.StoreChainProfileID
    join MainSecurity.dbo.StoreMaster sm on sm.StoreID = kim.StoreID
    leftjoin CustomerIdentificationTransaction cit on cit.CustomerIdentificationID
    = prt.CustomerIdentificationID and cit.CheckingAccountNumber1 =
20  prt.CheckingAccountNumber
    where
        prt.paymentrecordid = @PayID and
        prt.KioskPCID = @KioskPCID and
        bdt.CompanyAccountTypeID = 1 and
25  bdt.QBTed = 10
    order by bdt.kioskpcid, bdt.billdetailid

open PaymentRecordSet
fetch from PaymentRecordSet
30  into @OutputLine, @MultiPayBillID, @ReportedPayment
select @PaymentRecordFetchStatus = @@fetch_status

while @PaymentRecordFetchStatus = 0
begin
35  if @ReportedPayment > 0
    begin
        -- write actual record to file
        insert into ##HLP_PMT (OutputLine)
40  values
            (@OutputLine)

        select @FileTotal = @FileTotal + @ReportedPayment

```

```

end

-- don't forget to fetch the next one
fetch from PaymentRecordSet
5 into @OutputLine, @MultiPayBillID, @ReportedPayment
select @PaymentRecordFetchStatus = @@fetch_status

end

10 close PaymentRecordSet
deallocate PaymentRecordSet

-- don't forget to fetch the next one
fetch next from MultiPaySet
15 into @PayID, @KioskPCID
select @MultiPayFetchStatus = @@fetch_status

end

20 close MultiPaySet
deallocate MultiPaySet

insert into ##HLP_PMT (OutputLine)
values
25 ('-----')
insert into ##HLP_PMT (OutputLine)
values
(' Payment Total: ' + convert(char(12), @FileTotal))

30 -- print contents of temp table to disk file
declare @BCPCommand varchar(200)
select @BCPCommand = 'bcp "SELECT OutputLine FROM ##HLP_PMT order by id"
queryout "' + @FileName + "' /T /c'
35 EXEC master..xp_cmdshell @BCPCommand

select @BCPCommand = 'move ' + @FileName + '
\\ASKVTS01\Transmissions\outgoing\HLP\'
EXEC master..xp_cmdshell @BCPCommand
40

insert into
MainSecurity.dbo.JobRC

```

```
(DateTime, ReturnCode, JobTxt, TxtComment)
values
    (getdate(), 0, 'HLP PMT report', 'Report generated successfully.')
```

5

```
drop table ##HLP_PMT

end -- create proc
```

```

-- Kiosk Backup routine (reverses cleanup process)
-- depending on day of week
-
*****
5  -- RL 01/30/2001 : broke into separate companies, added day-of-week check
   -- RL 02/09/2001 : made from kiosk cleanup file

   use MainKiosk
   go
10  -- drop proc pr_kiosk_backup

   create proc pr_kiosk_backup
   as
15  begin

   declare @WeekDay varchar(10)

   select @WeekDay = datename(dw,getdate())
20  -- do not run this process if any records are in "to be processed/in process" status
   declare @TenCount int

   select @TenCount = isnull(count(*),0) from paymentrecordtransaction where qbted = 10
25  select @TenCount = @TenCount + isnull(count(*),0) from billdetailtransaction where qbted
   = 10
   select @TenCount = @TenCount + isnull(count(*),0) from cashvoucherissuedetailtransaction
   where qbted = 10
   select @TenCount = @TenCount + isnull(count(*),0) from returnedchecks where qbted = 10
30  if @TenCount <> 0
   begin
   insert into
       MainSecurity.dbo.JobRC
35  (DateTime, ReturnCode, JobTxt, TxtComment)
       values
       (getdate(), -1, 'Kiosk Backup', 'ABORTED: Kiosk Cleanup has not been run.')
   goto finish
   end
40  insert into
       MainSecurity.dbo.JobRC
       (DateTime, ReturnCode, JobTxt, TxtComment)

```

```

values
(getdate(), 0, 'Kiosk Backup', '** Kiosk Backup started. **')

-- HL&P bills
5      update BillDetailTransaction
      set QBTed = 10
      where (QBTed = 20) and CompanyAccountTypeID = 1
      insert into MainSecurity.dbo.JobRC
          (DateTime, ReturnCode, JobTxt, TxtComment)
10     values
          (getdate(), @@rowcount, 'Kiosk Backup', 'HL&P bill backup completed.')

-- HL&P NSF checks
      update ReturnedChecks
15     set QBTed = 10
      where (QBTed = 20) and rowguid in
          (select rc.rowguid
           from
20             ReturnedChecks rc
           join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
rc.KioskPCID = bp.KioskPCID
           join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
bp.KioskPCID = bdt.KioskPCID
           where
25             bdt.CompanyAccountTypeID = 1)
      insert into MainSecurity.dbo.JobRC
          (DateTime, ReturnCode, JobTxt, TxtComment)
      values
          (getdate(), @@rowcount, 'Kiosk Backup', 'HL&P NSF backup completed.')
30

-- Entex bills
-- do not process on Friday nights
if @WeekDay <> 'Friday'
35 begin
      update BillDetailTransaction
      set QBTed = 10
      where (QBTed = 20) and CompanyAccountTypeID in (2,9)
      insert into MainSecurity.dbo.JobRC
40     (DateTime, ReturnCode, JobTxt, TxtComment)
      values
          (getdate(), @@rowcount, 'Kiosk Backup', 'Entex bill backup completed.')

```

```

-- Entex NSF checks
    update ReturnedChecks
    set QBTed = 10
    where (QBTed = 20) and rowguid in
5      (select rc.rowguid
      from
          ReturnedChecks rc
          join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
rc.KioskPCID = bp.KioskPCID
10      join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
bp.KioskPCID = bdt.KioskPCID
      where
          bdt.CompanyAccountTypeID in (2,9))
    insert into MainSecurity.dbo.JobRC
15      (DateTime, ReturnCode, JobTxt, TxtComment)
    values
        (getdate(), @@rowcount, 'Kiosk Backup', 'Entex NSF backup completed.')

end
20
-- CHW bills
-- do not process on Sunday nights
if @WeekDay <> 'Sunday'
begin
25      update BillDetailTransaction
      set QBTed = 10
      where (QBTed = 20) and CompanyAccountTypeID = 4
      insert into MainSecurity.dbo.JobRC
          (DateTime, ReturnCode, JobTxt, TxtComment)
30      values
          (getdate(), @@rowcount, 'Kiosk Backup', 'CHW bill backup completed.')

-- CHW NSF checks
    update ReturnedChecks
35      set QBTed = 10
      where (QBTed = 20) and rowguid in
      (select rc.rowguid
      from
          ReturnedChecks rc
40      join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
rc.KioskPCID = bp.KioskPCID
      join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
bp.KioskPCID = bdt.KioskPCID

```

```

where
    bdt.CompanyAccountTypeID = 4)
insert into MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
5      values
        (getdate(), @@rowcount, 'Kiosk Backup', 'CHW NSF backup completed.')

end

10  -- TWC bills
    -- do not process on Sunday nights
    if @WeekDay <> 'Sunday'
    begin
15      update BillDetailTransaction
        set QBTed = 10
        where (QBTed = 20) and CompanyAccountTypeID = 10
        insert into MainSecurity.dbo.JobRC
            (DateTime, ReturnCode, JobTxt, TxtComment)
20      values
        (getdate(), @@rowcount, 'Kiosk Backup', 'TWC bill backup completed.')

    -- TWC NSF checks
        update ReturnedChecks
25      set QBTed = 10
        where (QBTed = 20) and rowguid in
            (select rc.rowguid
             from
                 ReturnedChecks rc
30      join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and
rc.KioskPCID = bp.KioskPCID
            join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and
bp.KioskPCID = bdt.KioskPCID
        where
35      bdt.CompanyAccountTypeID = 10)
        insert into MainSecurity.dbo.JobRC
            (DateTime, ReturnCode, JobTxt, TxtComment)
        values
            (getdate(), @@rowcount, 'Kiosk Backup', 'TWC NSF backup completed.')
40
    end

    -- SWB bills

```



-- do not process on Sunday nights

if @WeekDay <> 'Sunday'

begin

update CompanyAccountTypeMaster

set Notes = cast((cast(Notes as int) - 1) as varchar)

where CompanyAccountTypeID = 3

insert into MainSecurity.dbo.JobRC

(DateTime, ReturnCode, JobTxt, TxtComment)

values

(getdate(), @@rowcount, 'Kiosk Backup', 'SWB EDI control number changed.')

update BillDetailTransaction

set QBTed = 10

where (QBTed = 20) and CompanyAccountTypeID = 3

insert into MainSecurity.dbo.JobRC

(DateTime, ReturnCode, JobTxt, TxtComment)

values

(getdate(), @@rowcount, 'Kiosk Backup', 'SWB bill backup completed.')

update ReturnedChecks

set QBTed = 10

where (QBTed = 20) and rowguid in

(select rc.rowguid

from

ReturnedChecks rc

join BillPay bp on rc.PaymentRecordID = bp.PaymentRecordID and

rc.KioskPCID = bp.KioskPCID

join BillDetailTransaction bdt on bp.BillDetailID = bdt.BillDetailID and

bp.KioskPCID = bdt.KioskPCID

where

bdt.CompanyAccountTypeID = 3)

insert into MainSecurity.dbo.JobRC

(DateTime, ReturnCode, JobTxt, TxtComment)

values

(getdate(), @@rowcount, 'Kiosk Backup', 'SWB NSF backup completed.')

end

-- Chase payments

-- do not process on Sunday nights

if @WeekDay <> 'Sunday'

begin

```

update PaymentRecordTransaction
set QBTed = 10
where (QBTed = 20)
insert into MainSecurity.dbo.JobRC
5      (DateTime, ReturnCode, JobTxt, TxtComment)
values
      (getdate(), @@rowcount, 'Kiosk Backup', 'Chase payment backup completed.')

update CashVoucherIssueDetailTransaction
10    set QBTed = 10
      where (QBTed = 20)
insert into MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
values
15    (getdate(), @@rowcount, 'Kiosk Backup', 'Cash voucher backup completed.')

end

insert into MainSecurity.dbo.JobRC
20    (DateTime, ReturnCode, JobTxt, TxtComment)
values
      (getdate(), 0, 'Kiosk Backup', '** All bill backups completed. **')

finish:
25
return

end -- create proc
30

```

-- HL&P PMT

-

\*\*\*\*\*

-- AG & RL 11/25/2000 : insert detail records to HLP\_PMT table

5 -- RL 12/01/2000 : built for multi-pay processing

-- RL 12/02/2000 : modified to generate full PMT file (for all payments)

-- RL 01/03/2001 : batch on KioskPCID, report store info in file

-- RL 01/04/2001 : correct totals (over by svc chg, multi-pay)

-- RL 01/08/2001 : revise for MEMO file

10 -- RL 01/11/2001 : change multi-pay division method (from even split to proportional)

-- RL 01/25/2001 : change multi-pay reporting to match database change (AptPayAmt and ServiceAmt fields)

-- RL 01/25/2001 : write output to text file in its place on ASKVTS01

-- RL 01/27/2001 : use QBTEd value to choose records to process

15 -- RL 01/29/2001 : copy file, convert to stored procedures

-- RL 02/10/2001 : prevent double-subtraction of service change on mixed payments

-- RL 02/10/2001 : prevent reporting of numbers <=0 (\$1 payment - \$1 svc chg)

-- RL 02/15/2001 : rearrange figuring of totals (accumulate from detail instead of computing separately)

20 -- RL 02/26/2001 : make wire transfer write depend on file type

-- RL 03/29/2001 : add datetime values to join conditions

-- \*\*\*\*\*

use MainKiosk

25 go

drop proc pr\_HLP\_PMT

go

30 create proc pr\_HLP\_PMT  
as begin

-- overhead for file creation

create table ##HLP\_PMT

35 (id int identity, OutputLine varchar(200))

delete from ##HLP\_PMT

declare @FileName varchar(100)

-- is this a PMT or MEMO file?

40 declare @FileType varchar(4)

select @FileType = 'PMT'

--select @FileType = 'MEMO'

--\\ASKVTS01\Transmissions\outbound\HLP

```
select @FileName = 'c:\askHarris_HLP_' + @FileType + '_' + replace(convert(varchar,
getdate(), 106), ' ', '-') + '-' + right('0' + cast(datepart(hh,getdate()) as varchar(2)),2) + right('0' +
5 cast(datepart(mi,getdate()) as varchar(2)),2) + '.rpt'

-- values for each payment reported
declare @PayID int          -- payment id
declare @MultiPaySetCount int -- number of bills attached to payment
10 declare @KioskPCID int    -- kiosk where payment was entered
declare @TotalServiceCharge float -- service charge for the set of bills
declare @TotalAmountBilled float -- total amount of bills, including service charge
declare @TotalAmountPaid float    -- total amount of bills without service charge

15 -- parts of the output record
-- the multi-pay flag and sequence number go between these parts
declare @OutputPart1 varchar(200)
declare @OutputPart2 varchar(200)
declare @OutputPart3 varchar(200)
20 -- multi-pay flag parts
-- loop1 is the ASCII code for the alphabetic portion
-- loop2 if the numeric part
declare @mploop1 int
25 declare @mploop2 int
declare @MultiPayFlag char(3)
declare @StoreID char(9)

-- sequence number for lines in the PMT file
30 declare @SequenceNumber int
select @SequenceNumber = 0

-- accumulators for batch and file totals and counts
declare @BatchRecordCount int
35 declare @BatchTotal float
declare @FileTotal float
declare @BatchKiosk int

select
40     @BatchRecordCount = 0,
        @BatchTotal = 0,
        @FileTotal = 0
```

```

-- variables for wire transfer
declare @WireOffset int
if datename(dw,getdate()) in ('Thursday','Friday')
    select @WireOffset = 4
5  else
    select @WireOffset = 2

-- insert file header record
10  -- this record will be updated with the real file header later
insert into ##HLP_PMT (OutputLine)
values
(
'FILE HEADER'
15  + ' ' + 'ASK' + ' ' + '01' + ' ' +
case @FileType
when 'PMT' then 'PRD'
when 'MEMO' then 'MEM'
end
20  -- TST : Testing
-- PRD : Production
+ ' ' +
'00001' -- sequence number
+ ' ' +
25  cast(datepart(yy,getdate()) as char(4))+
right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)
)
30  select @SequenceNumber = @SequenceNumber + 1

-- get list of multi-pay payment records, with totals for each multi-pay
-- treat single-pay as a type of multipay, with MultiPaySetCount of 1
declare @MultiPayFetchStatus int
35  declare @MultiPayBillID int
declare @ReportedPayment float

declare MultiPaySet cursor for
select
40      prt.PaymentRecordID as PayID,
      prt.KioskPCID as KioskPCID,
      cast(upper(left(max(scp.StoreChainName),4)) as char(4))
      + ' ' + cast(right(max(sm.StoreName),3) as char(3)) as StoreID

```

```

from
    BillDetailTransaction bdt
    join BillPay bp on bp.KioskPCID = bdt.KioskPCID and bp.BillDetailID =
bdt.BillDetailID and bp.DateStamp = bdt.DateTimeofBillScan
5    join PaymentRecordTransaction prt on bp.KioskPCID = prt.KioskPCID and
prt.PaymentRecordID = bp.PaymentRecordid and prt.DateTimeofPayment = bp.DateStamp
    JOIN MainSecurity.dbo.KioskPCInfo kpc on bdt.KioskPCID = kpc.KioskPCID
    JOIN MainSecurity.dbo.KioskInfoMaster kim on kim.KioskInfoID = kpc.KioskID
    JOIN MainSecurity.dbo.StoreChainProfileMaster scp on kim.StoreChainProfileID =
10    scp.StoreChainProfileID
    JOIN MainSecurity.dbo.StoreMaster sm on kim.StoreID = sm.StoreID
where
    bdt.CompanyAccountTypeID = 1 and
    bdt.QBTed = 10
15 group by
    prt.paymentrecordid, prt.KioskPCID
order by
    prt.KioskPCID,
    count(distinct bp.billdetailid) desc
20
-- loop over payment records
-- this is the batch loop
open MultiPaySet
fetch from MultiPaySet
25 into @PayID, @KioskPCID, @StoreID
select @MultiPayFetchStatus = @@fetch_status

select @mploop1 = 65      -- ASCII for 'A'
30 select @BatchKiosk = 0

while @MultiPayFetchStatus = 0
begin
35    -- if this is a new batch, insert batch header record
    -- this record will be updated with the real batch header later
    if @BatchKiosk <> @KioskPCID
    begin
        -- if this is not the first batch, insert the batch total for the last batch
40        if @BatchKiosk <> 0
            update ##HLP_PMT
            set OutputLine = right('00000' + cast(@BatchRecordCount as
varchar(5)),5)

```



```

        case bdt.BillDetailID
            when @MultiPayBillID then right('000000000000' +
cast(cast(round(((isnull(bdt.AptPayAmt, prt.PaymentAmount)) * 100),0) as int) as
varchar(11)),11)
5         else right('000000000000' + cast(cast(round(((isnull(bdt.AptPayAmt,
prt.PaymentAmount) - bdt.ServiceAmt) * 100),0) as int) as varchar(11)),11)
            end
            + '' +
            case prt.paymenttypeid
10             when '1' then 'CA'
            when '2' then 'CK'
            end
            + ''
            as Part1,
15         -- MultiPayFlag goes between Part1 and Part2
            '' +
            right('0' + cast(datepart(hh,prt.DateTimeofPayment) as varchar(2)),2)+
            right('0' + cast(datepart(mi,prt.DateTimeofPayment) as varchar(2)),2)+
            right('0' + cast(datepart(ss,prt.DateTimeofPayment) as varchar(2)),2)
20         + '' + 'CIS ' + '' +
            @StoreID
            + '' +
            right('0000' + cast(@KioskPCID+8000 as varchar(4)), 4)
            + ''
25         as Part2,
            -- SequenceNumber goes between Part2 and Part3
            '' +
            cast(datepart(yy,prt.DateTimeofPayment) as varchar(4))+
            right('0' + cast(datepart(mm,prt.DateTimeofPayment) as varchar(2)),2)+
30         right('0' + cast(datepart(dd,prt.DateTimeofPayment) as varchar(2)),2)
            as Part3,
            bdt.BillDetailID as MultiPayBillID,
            case bdt.BillDetailID
            when @MultiPayBillID then isnull(bdt.AptPayAmt, prt.PaymentAmount)
35         else isnull(bdt.AptPayAmt, prt.PaymentAmount) - bdt.ServiceAmt
            end as ReportedPayment
from
            BillDetailTransaction bdt
            join BillPay bp on bp.KioskPCID = bdt.KioskPCID and bp.BillDetailID =
40         bdt.BillDetailID and bp.DateStamp = bdt.DateTimeofBillScan
            join PaymentRecordTransaction prt on bp.KioskPCID = prt.KioskPCID and
prt.PaymentRecordID = bp.PaymentRecordid and prt.DateTimeofPayment = bp.DateStamp
            join Mainsecurity.dbo.KioskPCInfo kpc on kpc.KioskPCID = bdt.KioskPCID

```



```

where
    prt.paymentrecordid = @PayID and
    prt.KioskPCID = @KioskPCID and
    bdt.CompanyAccountTypeID = 1 and
5    bdt.QBTed = 10
order by bdt.kioskpcid, bdt.billdetailid

open PaymentRecordSet
fetch from PaymentRecordSet
10    into @OutputPart1, @OutputPart2, @OutputPart3, @MultiPayBillID,
@ReportedPayment
select @PaymentRecordFetchStatus = @@fetch_status

select @mploop2 = 1
15
while @PaymentRecordFetchStatus = 0
begin
    if @ReportedPayment > 0
    20    begin
        select @SequenceNumber = @SequenceNumber + 1
        if @MultiPaySetCount > 1
            select @MultiPayFlag = char(@mploop1) + cast(@mploop2 as
varchar(2))
25    else
        select @MultiPayFlag = ' '

        -- write actual record to file
        insert into ##HLP_PMT (OutputLine)
        30    values
            (@OutputPart1 + @MultiPayFlag + @OutputPart2 + right('00000' +
cast(@SequenceNumber as varchar(5)), 5) + @OutputPart3)

        select @FileTotal = @FileTotal + @ReportedPayment
        select @BatchTotal = @BatchTotal + @ReportedPayment
        35    select @BatchRecordCount = @BatchRecordCount + 1
    end

    -- don't forget to fetch the next one
    40    fetch next from PaymentRecordSet
    into @OutputPart1, @OutputPart2, @OutputPart3, @MultiPayBillID,
@ReportedPayment
    select @PaymentRecordFetchStatus = @@fetch_status

```

```

        select @mploop2 = @mploop2 + 1

    end

5      close PaymentRecordSet
      deallocate PaymentRecordSet

      -- don't forget to fetch the next one
      fetch next from MultiPaySet
10     into @PayID, @KioskPCID, @StoreID
      select @MultiPayFetchStatus = @@fetch_status

      select @mploop1 = @mploop1 + 1

15    end

      close MultiPaySet
      deallocate MultiPaySet

20    -- write totals for the last batch
      update ##HLP_PMT
      set OutputLine = right('00000' + cast(@BatchRecordCount as varchar(5)),5)
                      + '      ' +
25      case @FileType
        when 'PMT' then '997'
        when 'MEMO' then '887'
        end
                      + '      ' +
30      right('000000000000000' + cast(cast(round((@BatchTotal * 100),0) as int) as
varchar(14)),14)
                      + substring(OutputLine , 13, 100)
      where OutputLine like 'BATCH HEADER%'

35    -- write totals for the file
      update ##HLP_PMT
      set OutputLine = right('00000' + cast(@SequenceNumber as varchar(5)),5)
                      + '      ' +
                      right('0' + cast(datepart(hh,getDate()) as varchar(2)),2)+
40      right('0' + cast(datepart(mi,getDate()) as varchar(2)),2)+
                      right('0' + cast(datepart(ss,getDate()) as varchar(2)),2)
                      + '      ' +
                      case @FileType

```

```

        when 'PMT' then '999'
        when 'MEMO' then '889'
    end
    + '      ' +
5      right('000000000000000' + cast(cast(round((@FileTotal * 100),0) as int) as
varchar(14)),14)
        + substring(OutputLine , 12, 100)
    where OutputLine like 'FILE HEADER%'

10  -- write wire transfer number to table
    if @FileType = 'PMT'
        insert into
            MainSecurity.dbo.WireTransfers
            (Payee, WireDate, WireAmount, Comment)
15      values
            ('HL&P', dateadd(day,@WireOffset,getdate()), @FileTotal, 'Generated ' +
convert(varchar,getdate(),101))

20  insert into ##HLP_PMT (OutputLine)
    values
        ('MMMM')

    -- print contents of temp table to disk file
25  declare @BCPCommand varchar(200)
    select @BCPCommand = 'bcp "SELECT OutputLine FROM ##HLP_PMT order by id"
queryout "' + @FileName + "' /T /c'
EXEC master..xp_cmdshell @BCPCommand
    select @BCPCommand = 'move ' + @FileName + '
30  \\ASKVTS01\Transmissions\outgoing\HLP\'
EXEC master..xp_cmdshell @BCPCommand

    insert into
        MainSecurity.dbo.JobRC
35      (DateTime, ReturnCode, JobTxt, TxtComment)
    values
        (getdate(), 0, 'HL&P PMT file', 'File generated successfully.')

    SELECT OutputLine FROM ##HLP_PMT order by id
40  drop table ##HLP_PMT

end -- create proc

```

```

-- HLP NSF report/invoice
-- Human-readable report and invoice
-
*****
5  -- RL 01/10/2001 : query written as example
   -- RL 01/19/2001 : store name/number added, with return code
   -- RL 01/25/2001 : send output to text file on ASKVTS01
   -- RL 01/27/2001 : use QBTed value to choose records to process
   -- RL 01/29/2001 : convert to stored procedure
10  -- RL 03/05/2001 : copied from Entex NSF report
   -- RL 03/29/2001 : add datetime values to join conditions

   use MainKiosk
   go
15
   drop proc pr_HLP_NSF_report
   go

   create proc pr_HLP_NSF_report
20  as begin

   -- overhead for file creation
   create table ##HLP_NSF
   (id int identity, OutputLine varchar(200))
25  delete from ##HLP_NSF
   declare @FileName varchar(100)

   select @FileName = 'c:\askHarris_HLP_NSFhuman_' + replace(convert(varchar, getdate(),
30  106), ' ', ',') + '-' + right('0' + cast(datepart(hh,getdate()) as varchar(2)),2) + right('0' +
   cast(datepart(mi,getdate()) as varchar(2)),2) + '.txt'

   declare @ReturnedAmount float
   declare @ReturnedTotal float
   declare @ReturnedCount int
35
   select  @ReturnedAmount = 0,
           @ReturnedTotal = 0,
           @ReturnedCount = 0

40  declare @OutputLine varchar(200)

   insert into ##HLP_NSF (OutputLine)
   values ("

```

```

insert into ##HLP_NSF (OutputLine)
values ('askHarris.com')
insert into ##HLP_NSF (OutputLine)
values ('RETURNED CHECK REPORT')
5 insert into ##HLP_NSF (OutputLine)
values ('for Reliant HL&P')
insert into ##HLP_NSF (OutputLine)
values ('Generated ' + convert(varchar, getdate()))
insert into ##HLP_NSF (OutputLine)
10 values ('')
insert into ##HLP_NSF (OutputLine)
values (' Payment                               Store
Store')
insert into ##HLP_NSF (OutputLine)
15 values (' Date      Account #      Amount      Name              TDL      Receipt #      Code
Chain              Name ')
insert into ##HLP_NSF (OutputLine)
values
-----')
20
declare NSFCursor cursor for
select
convert(char(10), prt.DateTimeofPayment, 101)
+ ' ' + -- as PaymentDate,
25 bdt.BillAccountNumber
+ ' ' + -- as AccountNumber
convert(char(12), prt.PaymentAmount - bdt.ServiceAmt, 0)
+ ' ' + -- as PaymentAmount,
cast(replace(cit.Name, '$', ' ') as char(30))
30 + ' ' + -- as CustomerName,
cit.IdentificationNumber
+ ' ' + -- as TDL,
cast(rdt.ReceiptNumber as char(14))
+ ' ' + -- as ReceiptNumber,
35 rc.ReturnCode
+ ' ' + -- as ReturnCode,
cast(scp.StoreChainName as char(20))
+ ' ' + -- as StoreChain,
sm.StoreName,
40 -- as StoreName
prt.PaymentAmount - bdt.ServiceAmt as ReturnedAmount
from
PaymentRecordTransaction prt

```

```

        join ReturnedChecks rc on prt.PaymentRecordID = rc.PaymentRecordID and
prt.KioskPCID = rc.KioskPCID
        join BillPay bp on prt.PaymentRecordID = bp.PaymentRecordID and prt.KioskPCID
= bp.KioskPCID and prt.DateTimeofPayment = bp.DateStamp
5        join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and bdt.KioskPCID
= bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
        join CustomerIdentificationTransaction cit on cit.CustomerIdentificationID =
prt.CustomerIdentificationID and cit.CheckingAccountNumber1 =
prt.CheckingAccountNumber
10        join ReceiptBill rb on rb.KioskPCID = bp.KioskPCID and rb.BillPayId = bp.BillPayID
and rb.DateStamp = bp.DateStamp
        join ReceiptDetailTransaction rdt on rdt.KioskPCID = rb.KioskPCID and
rdt.ReceiptDetailID = rb.ReceiptDetailID and rdt.DateTimeofReceiptIssue = rb.DateStamp
        join MainSecurity.dbo.KioskPCInfo kpc on kpc.KioskPCID = prt.KioskPCID
15        join MainSecurity.dbo.KioskInfoMaster kim on kpc.KioskID = kim.KioskInfoID
        join MainSecurity.dbo.StoreChainProfileMaster scp on kim.StoreChainProfileID =
scp.StoreChainProfileID
        join MainSecurity.dbo.StoreMaster sm on sm.StoreID = kim.StoreID
        join MainSecurity.dbo.AddressMaster am on am.StoreID = kim.StoreID
20 where
        bdt.CompanyAccountTypeId = 1 and
        rc.QBTed = 10
order by
        prt.DateTimeofPayment
25
open NSFCursor
fetch from NSFCursor into @OutputLine, @ReturnedAmount

while @@fetch_status = 0
30 begin
        select @ReturnedTotal = @ReturnedTotal + @ReturnedAmount
        select @ReturnedCount = @ReturnedCount + 1

        insert into ##HLP_NSF (OutputLine)
35 values (@OutputLine)
        fetch next from NSFCursor into @OutputLine, @ReturnedAmount
end

close NSFCursor
40 deallocate NSFCursor

insert into ##HLP_NSF (OutputLine)
values ("
```

```

insert into ##HLP_NSF (OutputLine)
values ('-----')
insert into ##HLP_NSF (OutputLine)
values ('Returned Checks      ' + convert(char(5), @ReturnedCount, 0) +
5  convert(char(12), cast(@ReturnedTotal as money), 0))
insert into ##HLP_NSF (OutputLine)
values ('Bank Fees           ' + convert(char(12), cast(@ReturnedCount * 2 as money),
0))
insert into ##HLP_NSF (OutputLine)
10  values ('askHarris Service Charge      ' + + convert(char(12), cast(@ReturnedCount as
money), 0))
insert into ##HLP_NSF (OutputLine)
values
('=====')
15  insert into ##HLP_NSF (OutputLine)
values ('TOTAL                        ' + convert(char(12), cast(@ReturnedTotal +
(@ReturnedCount * 3) as money), 0))
insert into ##HLP_NSF (OutputLine)
20  values ('')
insert into ##HLP_NSF (OutputLine)
values ('ACH Payment Instructions:')
insert into ##HLP_NSF (OutputLine)
values ('Account: askHarris.com')
25  insert into ##HLP_NSF (OutputLine)
values ('Bank   : Chase Bank of Texas')
insert into ##HLP_NSF (OutputLine)
values ('ABA #   : 111000609')
insert into ##HLP_NSF (OutputLine)
30  values ('Account #30802536134')

-- print contents of temp table to disk file
35  declare @BCPCommand varchar(200)
select @BCPCommand = 'bcp "SELECT OutputLine FROM ##HLP_NSF order by id"
queryout "' + @FileName + "' /T /c'
EXEC master..xp_cmdshell @BCPCommand

40  select @BCPCommand = 'move ' + @FileName + '
\\ASKVTS01\Transmissions\outgoing\HLP'
EXEC master..xp_cmdshell @BCPCommand

```

```

insert into
    MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
values
5      (getdate(), 0, 'HL&P NSF report', 'File generated successfully.')

drop table ##HLP_NSF

end -- create proc
10

```

10  
 11  
 12  
 13  
 14  
 15  
 16  
 17  
 18  
 19  
 20  
 21  
 22  
 23  
 24  
 25  
 26  
 27  
 28  
 29  
 30  
 31  
 32  
 33  
 34  
 35  
 36  
 37  
 38  
 39  
 40  
 41  
 42  
 43  
 44  
 45  
 46  
 47  
 48  
 49  
 50  
 51  
 52  
 53  
 54  
 55  
 56  
 57  
 58  
 59  
 60  
 61  
 62  
 63  
 64  
 65  
 66  
 67  
 68  
 69  
 70  
 71  
 72  
 73  
 74  
 75  
 76  
 77  
 78  
 79  
 80  
 81  
 82  
 83  
 84  
 85  
 86  
 87  
 88  
 89  
 90  
 91  
 92  
 93  
 94  
 95  
 96  
 97  
 98  
 99  
 100



```

-- HL&P NSF file
-
*****
-- RL 12/01/2000 : built for multi-pay processing
5  -- RL 12/02/2000 : modified to generate full PMT file (for all payments)
-- RL 01/03/2001 : batch on KioskPCID, report store info in file
-- RL 01/04/2001 : correct totals (over by svc chg, multi-pay)
-- RL 01/08/2001 : revise for MEMO file
-- RL 01/08/2001 : make NSF query from PMT/MEMO
10 -- RL 01/10/2001 : remove batching from NSF file
-- RL 01/25/2001 : change multi-pay reporting to match database change (AptPayAmt and
ServiceAmt)
-- RL 01/25/2001 : write output to text file on ASKVTS01
-- RL 01/27/2001 : use QBTed value to choose records to process
15 -- RL 01/29/2001 : convert to stored procedure
-- RL 03/29/2001 : add datetime values to join conditions
-- *****

use MainKiosk
20 go

drop proc pr_HLP_NSF
go

25 create proc pr_HLP_NSF
as begin

-- overhead for file creation
create table ##HLP_NSF
30 (id int identity, OutputLine varchar(200))
delete from ##HLP_NSF
declare @FileName varchar(100)

35

-- is this a PMT or MEMO file?
declare @FileType varchar(4)
select @FileType = 'NSF'

40 select @FileName = 'c:\askHarris_HLP_' + @FileType + '_' + replace(convert(varchar,
getdate(), 106), '-', '') + '-' + right('0' + cast(datepart(hh,getdate()) as varchar(2)),2) + right('0' +
cast(datepart(mi,getdate()) as varchar(2)),2) + '.rpt'

```

```

-- values for each payment reported
declare @PayID int -- payment id
declare @MultiPaySetCount int -- number of bills attached to payment
5 declare @KioskPCID int -- kiosk where payment was entered
declare @TotalServiceCharge float -- service charge for the set of bills
declare @TotalAmountBilled float -- total amount of bills, including service charge
declare @TotalAmountPaid float -- total amount of bills without service charge

10 -- parts of the output record
-- the multi-pay flag and sequence number go between these parts
declare @OutputPart1 varchar(200)
declare @OutputPart2 varchar(200)
declare @OutputPart3 varchar(200)

15 -- multi-pay flag parts
-- loop1 is the ASCII code for the alphabetic portion
-- loop2 if the numeric part
declare @mploop1 int
20 declare @mploop2 int
declare @MultiPayFlag char(3)

-- sequence number for lines in the PMT file
declare @SequenceNumber int
25 select @SequenceNumber = 1

-- figure totals for file header record
declare @RecordCount int
30 declare @TotalAmount float

select
    @RecordCount = isnull(count(*),0),
    @TotalAmount = isnull(sum(prt.PaymentAmount - bdt.ServiceAmt),0)
35 FROM
    BillDetailTransaction bdt
    JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
    bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
    JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId
40 and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
    JOIN ReturnedChecks rc on prt.KioskPCID = rc.KioskPCID and prt.PaymentRecordID
    = rc.PaymentRecordID
WHERE

```

```

        bdt.CompanyAccountTypeId = '1' and
        rc.QBTed = 10

-- adjust batch total for multi-pay payments
5  -- in the sum above, full payment is added for each bill, instead of partial payment
    select @TotalAmount = @TotalAmount - isnull(sum(foo),0)
    from
        (
            select
10         isnull(((max(prt.paymentamount))/(count(*)-1),0) as foo
            from
                BillDetailTransaction bdt
                JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
                bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
15         JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId =
                prt.PaymentRecordId and bp.KioskPCID = prt.KioskPCID and bp.DateStamp =
                prt.DateTimeofPayment
                JOIN ReturnedChecks rc on prt.KioskPCID = rc.KioskPCID and
                prt.PaymentRecordId = rc.PaymentRecordId
20         WHERE
                bdt.CompanyAccountTypeId = '1' and
                rc.QBTed = 10
            group by
                prt.paymentrecordid, prt.KioskPCID
25         having
                count(distinct bp.billdetailid) > 1
            ) as blah

30  -- generate file header record
    insert into ##HLP_NSF (OutputLine)
    values
    (
        right('00000' + cast(@RecordCount+2 as varchar(5)),5)
        + ' ' +
35         right('0' + cast(datepart(hh,getdate()) as varchar(2)),2)+
        right('0' + cast(datepart(mi,getdate()) as varchar(2)),2)+
        right('0' + cast(datepart(ss,getdate()) as varchar(2)),2)
        + ' ' +
        '999'
40         + ' ' +
        right('000000000000000' + cast(cast(round(((@TotalAmount * 100),0) as int) as
        varchar(14)),14)
        + ' ' + 'ASK' + ' ' + '01' + ' ' +

```

```

        'NSF'
        -- TST : Testing
        -- PRD : Production
        + ' ' +
5      right('00000' + cast(@SequenceNumber as varchar(5)),5)
        + ' ' +
        cast(datepart(yy,getDate()) as char(4))+
        right('0' + cast(datepart(mm,getDate()) as varchar(2)),2)+
        right('0' + cast(datepart(dd,getDate()) as varchar(2)),2)
10    )

    /*
    declare @BatchRecordCount int
    declare @BatchTotalAmount float
15    declare @BatchCursorFetchStatus int

    declare BatchCursor cursor for
    select
        PaymentRecordTransaction.KioskPCID,
20      cast(upper(left(LocalSecurity.dbo.StoreChainProfileMaster.StoreChainName,4)) as
        char(4))
        + ' ' + cast(right(LocalSecurity.dbo.StoreMaster.StoreName,3) as char(3)),
        count(*) as BatchRecordCount,
        sum(PaymentRecordTransaction.PaymentAmount + BillDetailTransaction.BillAmount
25    - BillDetailTransaction.AmountDue) as BatchTotalAmount
    FROM
        BillDetailTransaction
        JOIN BillPay ON BillDetailTransaction.BillDetailId = BillPay.BillDetailId
        JOIN PaymentRecordTransaction ON BillPay.PaymentRecordId =
30    PaymentRecordTransaction.PaymentRecordId
        JOIN PaymentType ON PaymentRecordTransaction.PaymentTypeId =
        PaymentType.PaymentTypeId
        JOIN LocalSecurity.dbo.KioskPCInfo on BillDetailTransaction.KioskPCID =
        LocalSecurity.dbo.KioskPCInfo.KioskPCID
35    JOIN LocalSecurity.dbo.KioskInfoMaster on
        LocalSecurity.dbo.KioskInfoMaster.KioskInfoID=LocalSecurity.dbo.KioskPCInfo.KioskID
        JOIN LocalSecurity.dbo.StoreChainProfileMaster on
        LocalSecurity.dbo.KioskInfoMaster.StoreChainProfileID =
        LocalSecurity.dbo.StoreChainProfileMaster.StoreChainProfileID
40    JOIN LocalSecurity.dbo.StoreMaster on LocalSecurity.dbo.KioskInfoMaster.StoreID
        = LocalSecurity.dbo.StoreMaster.StoreID
        JOIN ReturnedChecks on PaymentRecordTransaction.KioskPCID =
        ReturnedChecks.KioskPCID and PaymentRecordTransaction.PaymentRecordID =

```

```

ReturnedChecks.PaymentRecordID
WHERE
    BillDetailTransaction.CompanyAccountTypeID = '1'
group by
5     LocalSecurity.dbo.StoreChainProfileMaster.StoreChainName,
    LocalSecurity.dbo.StoreMaster.StoreName,
    PaymentRecordTransaction.KioskPCID
order by
    PaymentRecordTransaction.KioskPCID
10
open BatchCursor
fetch from BatchCursor
into @KioskPCID, @StoreID, @BatchRecordCount, @BatchTotalAmount
select @BatchCursorFetchStatus = @@fetch_status
15 */

select @SequenceNumber = @SequenceNumber + 1

20 -- adjust batch total for multi-pay payments
-- in the sum above, full payment is added for each bill, instead of partial payment
/*
select @BatchTotalAmount = @BatchTotalAmount - isNull(sum(foo),0)
from
25     (
        select
            (max(prt.paymentamount))/(count(*)-1) as foo
        from
            BillPay bp,
30         BillDetailTransaction bdt,
            PaymentRecordTransaction prt,
            ReturnedChecks rc
        where
            bdt.CompanyAccountTypeID = 1 and
35         bp.BillDetailID = bdt.BillDetailID and
            bp.PaymentRecordID = prt.PaymentRecordid and
            bp.kioskpcid = bdt.kioskpcid and
            bdt.kioskpcid = prt.kioskpcid and
            prt.KioskPCID = rc.KioskPCID and
40         prt.PaymentRecordID = rc.PaymentRecordID
        group by
            prt.paymentrecordid, prt.KioskPCID
        having

```

```

count(distinct bp.billdetailid) > 1
) as blah
*/

5  -- generate batch header record
insert into ##HLP_NSF (OutputLine)
values
(
    right('00000' + cast(@RecordCount+1 as varchar(5)),5)
    + '          ' +
10  '998'
    + '          ' +
    right('000000000000000' + cast(cast((@TotalAmount * 100) as int) as varchar(14)),14)
    + '          ' + '999' + ' ' +
    right('00000' + cast(@SequenceNumber as varchar(5)),5)
15  + ' ' +
    cast(datepart(yy,getDate()) as char(4))+
    right('0' + cast(datepart(mm,getDate()) as varchar(2)),2)+
    right('0' + cast(datepart(dd,getDate()) as varchar(2)),2)
)

20  -- get list of multi-pay payment records, with totals for each multi-pay
-- treat single-pay as a type of multipay, with MultiPaySetCount of 1
declare @StoreID char(9)
declare @MultiPayFetchStatus int
25  declare MultiPaySet cursor for
select
    prt.PaymentRecordID as PayID,
    prt.KioskPCID as KioskPCID,
    isnull(count(*),0) as MultiPaySetCount,
30  isnull(max(prt.PaymentAmount),0) as TotalAmountPaid,
    isnull(sum(bdt.ServiceAmt),0) as TotalServiceCharge,
    isnull(sum(bdt.BillAmount),0) as TotalAmountBilled
from
    BillDetailTransaction bdt
35  JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
    JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId
and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
    JOIN ReturnedChecks rc on prt.KioskPCID = rc.KioskPCID and prt.PaymentRecordID
40  = rc.PaymentRecordID
WHERE
    bdt.CompanyAccountTypeId = '1' and
    rc.QBTed = 10

```

```

group by
    prt.paymentrecordid, prt.KioskPCID
order by
    count(distinct bp.billdetailid) desc,
5     prt.KioskPCID
-- having
--     count(distinct bp.billdetailid) > 1

10  -- loop over payment records
open MultiPaySet
fetch from MultiPaySet
into @PayID, @KioskPCID, @MultiPaySetCount, @TotalAmountPaid, @TotalServiceCharge,
    @TotalAmountBilled
15  select @MultiPayFetchStatus = @@fetch_status

select @mploop1 = 65      -- ASCII for 'A'

while @MultiPayFetchStatus = 0
20  begin
    -- for each payment record, get bill records
    -- and data to report to PMT file
    declare @PaymentRecordFetchStatus int
    declare PaymentRecordSet cursor for
25  select
        '1 ' +
        left(bdt.BillAccountNumber, 12)
        + ' ' +
        right(bdt.BillAccountNumber, 1)
30  +
        ' ' +
        +
        -- if the set is not paid in full, divide payment equally between bills
        right('000000000000' + cast(cast(round(((isnull(bdt.AptPayAmt,
35  prt.PaymentAmount) - bdt.ServiceAmt) * 100),0) as int) as varchar(11)),11)
        + ' ' +
        '01'
        + ' '

        as Part1,
40  -- MultiPayFlag goes between Part1 and Part2
        '' +
        right('0' + cast(datepart(hh,prt.DateTimeofPayment) as varchar(2)),2)+
        right('0' + cast(datepart(mi,prt.DateTimeofPayment) as varchar(2)),2)+

```

```

right('0' + cast(datepart(ss,prt.DateTimeofPayment) as varchar(2)),2)
+ '' + 'CIS ' + '' +
cast(upper(left(scpc.StoreChainName,4)) as char(4))
+ ' ' + cast(right(sm.StoreName,3) as char(3))
5      + '' +
right('0000' + cast(@KioskPCID+8000 as varchar(4)), 4)
+ ''
      as Part2,
-- SequenceNumber goes between Part2 and Part3
10     '' +
cast(datepart(yy,prt.DateTimeofPayment) as varchar(4))+
right('0' + cast(datepart(mm,prt.DateTimeofPayment) as varchar(2)),2)+
right('0' + cast(datepart(dd,prt.DateTimeofPayment) as varchar(2)),2)
      as Part3
15     from
      BillDetailTransaction bdt
      JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
      JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId =
20     prt.PaymentRecordId and bp.KioskPCID = prt.KioskPCID and bp.DateStamp =
prt.DateTimeofPayment
      JOIN ReturnedChecks rc on prt.KioskPCID = rc.KioskPCID and
prt.PaymentRecordID = rc.PaymentRecordID
      JOIN MainSecurity.dbo.KioskPCInfo kpc on rc.KioskPCID = kpc.KioskPCID
25     JOIN MainSecurity.dbo.KioskInfoMaster kim on kim.KioskInfoID =
kpc.KioskID
      JOIN MainSecurity.dbo.StoreChainProfileMaster scp on
kim.StoreChainProfileID = scp.StoreChainProfileID
      JOIN MainSecurity.dbo.StoreMaster sm on kim.StoreID = sm.StoreID
30     WHERE
      bdt.CompanyAccountTypeId = '1' and
      prt.paymentrecordid = @PayID and
      prt.KioskPCID = @KioskPCID and
      rc.QBTed = 10
35     order by bdt.billdetailid

open PaymentRecordSet
fetch from PaymentRecordSet
into @OutputPart1, @OutputPart2, @OutputPart3
40     select @PaymentRecordFetchStatus = @@fetch_status

select @mploop2 = 1

```



```

while @PaymentRecordFetchStatus = 0
begin

    select @SequenceNumber = @SequenceNumber + 1
    if @MultiPaySetCount > 1
        select @MultiPayFlag = char(@mploop1) + cast(@mploop2 as
5      varchar(2))
    else
        select @MultiPayFlag = ' '
10

    -- write actual record to file
    insert into ##HLP_NSF (OutputLine)
    values (@OutputPart1 + @MultiPayFlag + @OutputPart2 + right('00000' +
cast(@SequenceNumber as varchar(5)), 5)+ @OutputPart3)
15

    -- don't forget to fetch the next one
    fetch next from PaymentRecordSet
    into @OutputPart1, @OutputPart2, @OutputPart3
    select @PaymentRecordFetchStatus = @@fetch_status
20    select @mploop2 = @mploop2 + 1

    end

    close PaymentRecordSet
    deallocate PaymentRecordSet
25

    -- don't forget to fetch the next one
    fetch next from MultiPaySet
    into @PayID, @KioskPCID, @MultiPaySetCount, @TotalAmountPaid,
30 @TotalServiceCharge, @TotalAmountBilled
    select @MultiPayFetchStatus = @@fetch_status

    select @mploop1 = @mploop1 + 1

35 end

close MultiPaySet
deallocate MultiPaySet

40 insert into ##HLP_NSF (OutputLine)
values
('MMMM')

```

```

-- print contents of temp table to disk file
declare @BCPCommand varchar(200)
select @BCPCommand = 'bcp "SELECT OutputLine FROM ##HLP_NSF order by id"
queryout "' + @FileName + "' /T /c'
5 EXEC master..xp_cmdshell @BCPCommand
select @BCPCommand = 'move ' + @FileName + '
\\ASKVTS01\Transmissions\outgoing\HLP\'
EXEC master..xp_cmdshell @BCPCommand

10 insert into
    MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
values
15 (getdate(), 0, 'HL&P NSF file', 'File generated successfully.')

drop table ##HLP_NSF

end -- create proc
20

```

```
-- Entex PMT report
-- Human-readable report of Entex payments
```

```
*****
```

```
5  -- RL 01/10/2001 : query written as example
   -- RL 01/25/2001 : send output to text file on ASKVTS01
   -- RL 01/27/2001 : use QBTEd value to choose records to process
   -- RL 01/29/2001 : convert to stored procedure
   -- RL 02/02/2001 : copy from Entex NSF report
10  -- RL 02/03/2001 : correct for mixed payments (one payment, one service charge)
   -- RL 02/10/2001 : prevent reporting of numbers <=0 ($1 payment - $1 svc chg)
   -- RL 02/16/2001 : only report current day's payments, fix Customer info
   -- RL 02/23/2001 : add grand total at bottom of report
   -- RL 03/29/2001 : add datetime values to join conditions

15  use MainKiosk
   go

   drop proc pr_Entex_PMT_report
20  go

   create proc pr_Entex_PMT_report
   as begin

25  -- overhead for file creation
   create table ##ENTEX_PMT
   (id int identity, OutputLine varchar(200))
   delete from ##ENTEX_PMT
   declare @FileName varchar(100)

30  select @FileName = 'c:\askHarris_Entex_PMTHuman_' + replace(convert(varchar, getdate(),
   106), ' ', ',') + '-' + right('0' + cast(datepart(hh, getdate()) as varchar(2)), 2) + right('0' +
   cast(datepart(mi, getdate()) as varchar(2)), 2) + '.txt'

35  declare @OutputLine varchar(200)
   declare @DetailAmount money
   declare @TotalAmount money

40  select @DetailAmount = 0, @TotalAmount = 0

   insert into ##ENTEX_PMT (OutputLine)
   values ('')
```

```

insert into ##ENTEX_PMT (OutputLine)
values ('askHarris.com')
insert into ##ENTEX_PMT (OutputLine)
values ('PAYMENT REPORT')
5 insert into ##ENTEX_PMT (OutputLine)
values ('for Reliant Entex')
insert into ##ENTEX_PMT (OutputLine)
values ('Generated ' + convert(varchar, getdate()))
insert into ##ENTEX_PMT (OutputLine)
10 values ('')
insert into ##ENTEX_PMT (OutputLine)
values (' Payment                               Store          Store')
insert into ##ENTEX_PMT (OutputLine)
values (' Date   Account #   Amount   Name           TDL   Receipt #   Chain
15         Name ')
insert into ##ENTEX_PMT (OutputLine)
values
-----')

20 declare NSFCursor cursor for
select
    convert(char(10), max(prt.DateTimeofPayment), 101)
    + ' ' + -- as PaymentDate,
    max(bdt.BillAccountNumber)
25 + ' ' + -- as AccountNumber
    convert(char(12), sum(prt.PaymentAmount) - max(bdt.ServiceAmt), 0)
    + ' ' + -- as PaymentAmount,
    cast(replace(max(isnull(cit.Name, ' [Cash Payment]')), '$', ' ') as char(30))
    + ' ' + -- as CustomerName,
30 max(isnull(cit.IdentificationNumber, '    '))
    + ' ' + -- as TDL,
    cast(max(rdt.ReceiptNumber) as char(14))
    + ' ' + -- as ReturnCode,
    cast(max(scp.StoreChainName) as char(20))
35 + ' ' + -- as StoreChain,
    max(sm.StoreName),
    -- as StoreName
    sum(prt.PaymentAmount) - max(bdt.ServiceAmt) as DetailAmount
from
40 PaymentRecordTransaction prt
    join BillPay bp on prt.PaymentRecordID = bp.PaymentRecordID and prt.KioskPCID
    = bp.KioskPCID and prt.DateTimeofPayment = bp.DateStamp
    join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and bdt.KioskPCID

```

```

= bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
    left join CustomerIdentificationTransaction cit on cit.CustomerIdentificationID =
prt.CustomerIdentificationID and cit.CheckingAccountNumber1 =
prt.CheckingAccountNumber
5    join ReceiptBill rb on rb.KioskPCID = bp.KioskPCID and rb.BillPayId = bp.BillPayID
and rb.DateStamp = bp.DateStamp
    join ReceiptDetailTransaction rdt on rdt.KioskPCID = rb.KioskPCID and
rdt.ReceiptDetailID = rb.ReceiptDetailID and rdt.DateTimeofReceiptIssue = rb.DateStamp
    join MainSecurity.dbo.KioskPCInfo kpc on kpc.KioskPCID = prt.KioskPCID
10    join MainSecurity.dbo.KioskInfoMaster kim on kpc.KioskID = kim.KioskInfoID
    join MainSecurity.dbo.StoreChainProfileMaster scp on kim.StoreChainProfileID =
scp.StoreChainProfileID
    join MainSecurity.dbo.StoreMaster sm on sm.StoreID = kim.StoreID
    join MainSecurity.dbo.AddressMaster am on am.StoreID = kim.StoreID
15 where
    bdt.CompanyAccountTypeID in (2, 9) and
    bdt.QBTed = 10
group by bdt.BillDetailID, bdt.KioskPCID
having (sum(prt.PaymentAmount) - max(bdt.ServiceAmt)) > 0
20 order by
    max(prt.DateTimeofPayment)

open NSFCursor
fetch from NSFCursor into @OutputLine, @DetailAmount
25 while @@fetch_status = 0
begin
    insert into ##ENTEX_PMT (OutputLine)
    values (@OutputLine)
30    select @TotalAmount = @TotalAmount + @DetailAmount
    fetch next from NSFCursor into @OutputLine, @DetailAmount
end

close NSFCursor
35 deallocate NSFCursor

insert into ##ENTEX_PMT (OutputLine)
values

40 ('=====
=====')

insert into ##ENTEX_PMT (OutputLine)
values ('Payment Total ' + convert(char(12), @TotalAmount, 0) )

```

```

-- print contents of temp table to disk file
declare @BCPCommand varchar(200)
select @BCPCommand = 'bcp "SELECT OutputLine FROM ##ENTEX_PMT order by id"
queryout "' + @FileName + "' /T /c'
5 EXEC master..xp_cmdshell @BCPCommand

select @BCPCommand = 'move ' + @FileName + '
\\ASKVTS01\Transmissions\outgoing\Entex\'
EXEC master..xp_cmdshell @BCPCommand
10

insert into
    MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
values
15     (getdate(), 0, 'Entex PMT report', 'File generated successfully.')

drop table ##ENTEX_PMT

end -- create proc
20

```

-- HL&P MEMO

-  
\*\*\*\*\*

-- AG & RL 11/25/2000 : insert detail records to HLP\_PMT table

5 -- RL 12/01/2000 : built for multi-pay processing

-- RL 12/02/2000 : modified to generate full PMT file (for all payments)

-- RL 01/03/2001 : batch on KioskPCID, report store info in file

-- RL 01/04/2001 : correct totals (over by svc chg, multi-pay)

-- RL 01/08/2001 : revise for MEMO file

10 -- RL 01/11/2001 : change multi-pay division method (from even split to proportional)

-- RL 01/25/2001 : change multi-pay reporting to match database change (AptPayAmt and ServiceAmt fields)

-- RL 01/25/2001 : write output to text file in its place on ASKVTS01

-- RL 01/27/2001 : use QBTEd value to choose records to process

15 -- RL 01/29/2001 : copy file, convert to stored procedures

-- RL 02/10/2001 : prevent double-subtraction of service change on mixed payments

-- RL 02/10/2001 : prevent reporting of numbers <=0 (\$1 payment - \$1 svc chg)

-- RL 02/15/2001 : rearrange figuring of totals (accumulate from detail instead of computing separately)

20 -- RL 02/26/2001 : make wire transfer write depend on file type

-- RL 02/26/2001 : re-copied file from PMT generator

-- RL 03/29/2001 : add datetime values to join conditions

-- \*\*\*\*\*

25 use MainKiosk

go

drop proc pr\_HLP\_MEMO

go

30 create proc pr\_HLP\_MEMO  
as begin

-- overhead for file creation

35 create table ##HLP\_PMT  
(id int identity, OutputLine varchar(200))  
delete from ##HLP\_PMT  
declare @FileName varchar(100)

40 -- is this a PMT or MEMO file?

declare @FileType varchar(4)

--select @FileType = 'PMT'

select @FileType = 'MEMO'

--\\ASKVTS01\Transmissions\outbound\HLP

select @FileName = 'c:\askHarris\_HLP\_' + @FileType + '\_' + replace(convert(varchar,  
getdate(), 106), ' ', '-') + '-' + right('0' + cast(datepart(hh,getdate()) as varchar(2)),2) + right('0' +  
5 cast(datepart(mi,getdate()) as varchar(2)),2) + '.rpt'

-- values for each payment reported  
declare @PayID int -- payment id  
declare @MultiPaySetCount int -- number of bills attached to payment  
10 declare @KioskPCID int -- kiosk where payment was entered  
declare @TotalServiceCharge float -- service charge for the set of bills  
declare @TotalAmountBilled float -- total amount of bills, including service charge  
declare @TotalAmountPaid float -- total amount of bills without service charge

15 -- parts of the output record  
-- the multi-pay flag and sequence number go between these parts  
declare @OutputPart1 varchar(200)  
declare @OutputPart2 varchar(200)  
declare @OutputPart3 varchar(200)  
20 -- multi-pay flag parts  
-- loop1 is the ASCII code for the alphabetic portion  
-- loop2 if the numeric part  
declare @mploop1 int  
25 declare @mploop2 int  
declare @MultiPayFlag char(3)  
declare @StoreID char(9)

-- sequence number for lines in the PMT file  
30 declare @SequenceNumber int  
select @SequenceNumber = 0

-- accumulators for batch and file totals and counts  
declare @BatchRecordCount int  
35 declare @BatchTotal float  
declare @FileTotal float  
declare @BatchKiosk int

select  
40 @BatchRecordCount = 0,  
@BatchTotal = 0,  
@FileTotal = 0



```

-- variables for wire transfer
declare @WireOffset int
if datename(dw,getdate()) in ('Thursday','Friday')
    select @WireOffset = 4
5  else
    select @WireOffset = 2

-- insert file header record
10  -- this record will be updated with the real file header later
insert into ##HLP_PMT (OutputLine)
values
(
'FILE HEADER'
15  + ' ' + 'ASK' + ' ' + '01' + ' ' +
case @FileType
when 'PMT' then 'PRD'
when 'MEMO' then 'MEM'
end
20  -- TST : Testing
-- PRD : Production
+ ' ' +
'00001' -- sequence number
+ ' ' +
25  cast(datepart(yy,getdate()) as char(4))+
right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)
)

30  select @SequenceNumber = @SequenceNumber + 1

-- get list of multi-pay payment records, with totals for each multi-pay
-- treat single-pay as a type of multipay, with MultiPaySetCount of 1
declare @MultiPayFetchStatus int
35  declare @MultiPayBillID int
declare @ReportedPayment float

declare MultiPaySet cursor for
select
40      prt.PaymentRecordID as PayID,
      prt.KioskPCID as KioskPCID,
      cast(upper(left(max(scp.StoreChainName),4)) as char(4))
      + ' ' + cast(right(max(sm.StoreName),3) as char(3)) as StoreID

```

```

from
    BillDetailTransaction bdt
    join BillPay bp on bp.KioskPCID = bdt.KioskPCID and bp.BillDetailID =
bdt.BillDetailID and bp.DateStamp = bdt.DateTimeofBillScan
5    join PaymentRecordTransaction prt on bp.KioskPCID = prt.KioskPCID and
prt.PaymentRecordID = bp.PaymentRecordid and prt.DateTimeofPayment = bp.DateStamp
    JOIN MainSecurity.dbo.KioskPCInfo kpc on bdt.KioskPCID = kpc.KioskPCID
    JOIN MainSecurity.dbo.KioskInfoMaster kim on kim.KioskInfoID = kpc.KioskID
    JOIN MainSecurity.dbo.StoreChainProfileMaster scp on kim.StoreChainProfileID =
10 scp.StoreChainProfileID
    JOIN MainSecurity.dbo.StoreMaster sm on kim.StoreID = sm.StoreID
where
    bdt.CompanyAccountTypeID = 1 and
    bdt.QBTed = 10
15 group by
    prt.paymentrecordid, prt.KioskPCID
order by
    prt.KioskPCID,
    count(distinct bp.billdetailid) desc
20
-- loop over payment records
-- this is the batch loop
open MultiPaySet
fetch from MultiPaySet
25 into @PayID, @KioskPCID, @StoreID
select @MultiPayFetchStatus = @@fetch_status

select @mploop1 = 65      -- ASCII for 'A'

30 select @BatchKiosk = 0

while @MultiPayFetchStatus = 0
begin

35     -- if this is a new batch, insert batch header record
    -- this record will be updated with the real batch header later
    if @BatchKiosk <> @KioskPCID
    begin
        -- if this is not the first batch, insert the batch total for the last batch
40         if @BatchKiosk <> 0
            update ##HLP_PMT
            set OutputLine = right('00000' + cast(@BatchRecordCount as
varchar(5)),5)

```

```

+ '      ' +
case @FileType
when 'PMT' then '997'
when 'MEMO' then '887'
5      end
+ '      ' +
right('00000000000000' + cast(cast(round((@BatchTotal
* 100),0) as int) as varchar(14)),14)
+ substring(OutputLine , 13, 100)
10      where OutputLine like 'BATCH HEADER%'

select @BatchKiosk = @KioskPCID
select @BatchTotal = 0
select @BatchRecordCount = 1
15      select @SequenceNumber = @SequenceNumber + 1
insert into ##HLP_PMT (OutputLine)
values
(      'BATCH HEADER' +
'      ' + @StoreID + ' ' + right('0000' + cast(@BatchKiosk+8000 as
20      varchar(4)), 4) + ' ' +
right('00000' + cast(@SequenceNumber as varchar(5)),5)
+ ' ' +
cast(datepart(yy,getDate()) as char(4))+
right('0' + cast(datepart(mm,getDate()) as varchar(2)),2)+
25      right('0' + cast(datepart(dd,getDate()) as varchar(2)),2)
)
end

-- for each payment record, get bill records
30      -- and data to report to PMT file
declare @PaymentRecordFetchStatus int
declare PaymentRecordSet cursor for
select
'1 ' +
35      left(bdt.BillAccountNumber, 12)
+ ' ' +
right(bdt.BillAccountNumber, 1)
+
case @FileType
40      when 'PMT' then ' '
when 'MEMO' then ' UN '
end
+

```

```

        case bdt.BillDetailID
            when @MultiPayBillID then right('000000000000' +
cast(cast(round(((isnull(bdt.AptPayAmt, prt.PaymentAmount)) * 100),0) as int) as
varchar(11)),11)
5         else right('000000000000' + cast(cast(round(((isnull(bdt.AptPayAmt,
prt.PaymentAmount) - bdt.ServiceAmt) * 100),0) as int) as varchar(11)),11)
            end
            + '' +
            case prt.paymenttypeid
10         when '1' then 'CA'
            when '2' then 'CK'
            end
            + ''

            as Part1,
15         -- MultiPayFlag goes between Part1 and Part2
            '' +
            right('0' + cast(datepart(hh,prt.DateTimeofPayment) as varchar(2)),2)+
            right('0' + cast(datepart(mi,prt.DateTimeofPayment) as varchar(2)),2)+
            right('0' + cast(datepart(ss,prt.DateTimeofPayment) as varchar(2)),2)
20         + '' + 'CIS ' + '' +
            @StoreID
            + '' +
            right('0000' + cast(@KioskPCID+8000 as varchar(4)), 4)
            + ''

25         as Part2,
            -- SequenceNumber goes between Part2 and Part3
            '' +
            cast(datepart(yy,prt.DateTimeofPayment) as varchar(4))+
            right('0' + cast(datepart(mm,prt.DateTimeofPayment) as varchar(2)),2)+
            right('0' + cast(datepart(dd,prt.DateTimeofPayment) as varchar(2)),2)
30         as Part3,
            bdt.BillDetailID as MultiPayBillID,
            case bdt.BillDetailID
            when @MultiPayBillID then isnull(bdt.AptPayAmt, prt.PaymentAmount)
35         else isnull(bdt.AptPayAmt, prt.PaymentAmount) - bdt.ServiceAmt
            end as ReportedPayment

        from
            BillDetailTransaction bdt
            join BillPay bp on bp.KioskPCID = bdt.KioskPCID and bp.BillDetailID =
40         bdt.BillDetailID and bp.DateStamp = bdt.DateTimeofBillScan
            join PaymentRecordTransaction prt on bp.KioskPCID = prt.KioskPCID and
prt.PaymentRecordID = bp.PaymentRecordid and prt.DateTimeofPayment = bp.DateStamp
            join Mainsecurity.dbo.KioskPCInfo kpc on kpc.KioskPCID = bdt.KioskPCID

```

```

where
    prt.paymentrecordid = @PayID and
    prt.KioskPCID = @KioskPCID and
    bdt.CompanyAccountTypeID = 1 and
5    bdt.QBTed = 10
order by bdt.kioskpcid, bdt.billdetailid

open PaymentRecordSet
fetch from PaymentRecordSet
10    into @OutputPart1, @OutputPart2, @OutputPart3, @MultiPayBillID,
    @ReportedPayment
select @PaymentRecordFetchStatus = @@fetch_status

select @mploop2 = 1
15
while @PaymentRecordFetchStatus = 0
begin
    if @ReportedPayment > 0
    20    begin
        select @SequenceNumber = @SequenceNumber + 1
        if @MultiPaySetCount > 1
            select @MultiPayFlag = char(@mploop1) + cast(@mploop2 as
    varchar(2))
25        else
            select @MultiPayFlag = ' '

        -- write actual record to file
        insert into ##HLP_PMT (OutputLine)
        30        values
            (@OutputPart1 + @MultiPayFlag + @OutputPart2 + right('00000' +
            cast(@SequenceNumber as varchar(5)), 5) + @OutputPart3)

        select @FileTotal = @FileTotal + @ReportedPayment
        select @BatchTotal = @BatchTotal + @ReportedPayment
        35        select @BatchRecordCount = @BatchRecordCount + 1
    end

    -- don't forget to fetch the next one
    40    fetch next from PaymentRecordSet
    into @OutputPart1, @OutputPart2, @OutputPart3, @MultiPayBillID,
    @ReportedPayment
    select @PaymentRecordFetchStatus = @@fetch_status

```

```

        select @mploop2 = @mploop2 + 1

    end

5      close PaymentRecordSet
      deallocate PaymentRecordSet

      -- don't forget to fetch the next one
      fetch next from MultiPaySet
10     into @PayID, @KioskPCID, @StoreID
      select @MultiPayFetchStatus = @@fetch_status

      select @mploop1 = @mploop1 + 1

15    end

      close MultiPaySet
      deallocate MultiPaySet

20    -- write totals for the last batch
      update ##HLP_PMT
      set OutputLine = right('00000' + cast(@BatchRecordCount as varchar(5)),5)
                      + '      ' +
25          case @FileType
              when 'PMT' then '997'
              when 'MEMO' then '887'
              end
                      + '      ' +
30          right('000000000000000' + cast(cast(round(((@BatchTotal * 100),0) as int) as
varchar(14)),14)
                      + substring(OutputLine , 13, 100)
      where OutputLine like 'BATCH HEADER%'

35    -- write totals for the file
      update ##HLP_PMT
      set OutputLine = right('00000' + cast(@SequenceNumber as varchar(5)),5)
                      + '      ' +
                      right('0' + cast(datepart(hh,getDate()) as varchar(2)),2)+
40          right('0' + cast(datepart(mi,getDate()) as varchar(2)),2)+
                      right('0' + cast(datepart(ss,getDate()) as varchar(2)),2)
                      + '      ' +
                      case @FileType

```

```

        when 'PMT' then '999'
        when 'MEMO' then '889'
    end
    + '      ' +
5      right('000000000000000' + cast(cast(round(((@FileTotal * 100),0) as int) as
varchar(14)),14)
        + substring(OutputLine , 12, 100)
    where OutputLine like 'FILE HEADER%'

10  -- write wire transfer number to table
    if @FileType = 'PMT'
        insert into
            MainSecurity.dbo.WireTransfers
            (Payee, WireDate, WireAmount, Comment)
15      values
            ('HL&P', dateadd(day,@WireOffset,getdate()), @FileTotal, 'Generated ' +
convert(varchar,getdate(),101))

20  insert into ##HLP_PMT (OutputLine)
    values
    ('MMMM')

    -- print contents of temp table to disk file
25  declare @BCPCommand varchar(200)
    select @BCPCommand = 'bcp "SELECT OutputLine FROM ##HLP_PMT order by id"
queryout "' + @FileName + "' /T /c'
EXEC master..xp_cmdshell @BCPCommand
    select @BCPCommand = 'move ' + @FileName + '
30  \\ASKVTS01\\Transmissions\\outgoing\\HLP\'
EXEC master..xp_cmdshell @BCPCommand

    insert into
        MainSecurity.dbo.JobRC
35      (DateTime, ReturnCode, JobTxt, TxtComment)
    values
        (getdate(), 0, 'HL&P ' + @FileType + ' file', 'File generated successfully.')

    SELECT OutputLine FROM ##HLP_PMT order by id
40  drop table ##HLP_PMT

    end -- create proc

```

```

-- Entex PMT file
-
*****
-- AG & RL 11/25/2000 : insert detail records into Entex_PMT table
5  -- RL 11/29/2000 : no insertion, wrap in header/footer
-- RL 12/11/2000 : switch to print statements instead of selecting unions, print one batch per
kiosk
-- RL 01/08/2001 : change terminal id stuff at end of batch header
-- RL 01/25/2001 : send output to text file on ASKVTS01
10 -- RL 01/27/2001 : use QBTEd value to choose records to process
-- RL 01/29/2001 : convert to stored procedure
-- RL 01/31/2001 : correct for two-part pay (was applying service charge on each pmt, not each
bill)
-- RL 02/09/2001 : prevent reporting of zero payments
15 -- RL 02/10/2001 : prevent reporting of numbers <=0 ($1 payment - $1 svc chg)
-- RL 03/29/2001 : add datetime values to join conditions

use MainKiosk
go
20
drop proc pr_Entex_PMT
go

create proc pr_Entex_PMT
25 as begin

-- overhead for file creation
create table ##ENTEX_PMT
(id int identity, OutputLine varchar(200))
30 delete from ##ENTEX_PMT
declare @FileName varchar(100)

select @FileName = 'c:\askHarris_Entex_PMT_' + replace(convert(varchar, getdate(), 106), '
', ',') + '-' + right('0' + cast(datepart(hh,getdate()) as varchar(2)),2) + right('0' +
35 cast(datepart(mi,getdate()) as varchar(2)),2) + '.rpt'

-- variables for wire transfer at end of procedure
declare @WireOffset int
if datename(dw,getdate()) in ('Thursday','Friday')
40 select @WireOffset = 4
else
select @WireOffset = 2

```



```

declare @BatchTotal float
declare @RecordCount int
declare @BatchKiosk int
declare @OutputLine varchar(200)
5 declare @FileTotal float
declare @FileRecordCount int
declare @BatchCount int

-- totals for file trailer record
10 select
    @FileTotal = 0,
    @FileRecordCount = 0

-- find batches for looping
15 -- batch by kiosk
declare BatchCursor cursor for
select distinct KioskPCID
from
    BillDetailTransaction
20 WHERE
    (BillDetailTransaction.CompanyAccountId = '2' OR
    BillDetailTransaction.CompanyAccountId = '9') and
    BillDetailTransaction.QBTed = 10
order by
25 KioskPCID

open BatchCursor
fetch from BatchCursor into @BatchKiosk

30 select @BatchCount = 0

while @@fetch_status = 0
begin

35 select @BatchCount = @BatchCount + 1

-- figure numbers for header/footer
select @BatchTotal = sum(field1), @RecordCount = count(*)
from (select
40 isnull((sum(PaymentRecordTransaction.PaymentAmount) -
max(BillDetailTransaction.ServiceAmt)),0) as field1
FROM
    BillDetailTransaction

```

```

        JOIN BillPay ON BillDetailTransaction.BillDetailId = BillPay.BillDetailId and
        BillPay.KioskPCID = BillDetailTransaction.KioskPCID and BillPay.DateStamp =
        BillDetailTransaction.DateTimeofBillScan
        JOIN PaymentRecordTransaction ON BillPay.PaymentRecordId =
5 PaymentRecordTransaction.PaymentRecordId and BillPay.KioskPCID =
PaymentRecordTransaction.KioskPCID and BillPay.DateStamp =
PaymentRecordTransaction.DateTimeofPayment
    WHERE
        (BillDetailTransaction.CompanyAccountTypeId = '2' OR
10 BillDetailTransaction.CompanyAccountTypeId = '9') and
        BillDetailTransaction.KioskPCID = @BatchKiosk and
        BillDetailTransaction.QBTed = 10 and
        PaymentRecordTransaction.PaymentAmount > 0
    group by BillDetailTransaction.billdetailid
15 having (sum(paymentrecordtransaction.PaymentAmount) -
max(billdetailtransaction.ServiceAmt)) > 0) foo

    select @FileTotal = @FileTotal + @BatchTotal

20 -- Record Type B
    -- Batch Header
    insert into ##ENTEX_PMT (OutputLine)
    values
    (
25 'B'
        + -- as RecordTypeCode,
        '40'
        + -- as DivisionCode,
        '030'
        + -- as OfficeCode,
30 '00096'
        + -- as BatchNumber,
        right('0000000000' + cast(cast(round((@BatchTotal * 100),0) as int) as
varchar(10)),10)
        + -- as BatchTotal,
35 right('0' + cast(datepart(mm,dateadd(day, 1, getdate())) as varchar(2)),2) +
        right('0' + cast(datepart(dd,dateadd(day, 1, getdate())) as varchar(2)),2) +
        cast(right(datepart(yy,dateadd(day, 1, getdate())) as char(2))
        + -- as BatchDate,
        right('0' + cast(datepart(mm,dateadd(day, 1, getdate())) as varchar(2)),2) +
40 right('0' + cast(datepart(dd,dateadd(day, 1, getdate())) as varchar(2)),2) +
        cast(right(datepart(yy,dateadd(day, 1, getdate())) as char(2))
        + -- as ProcessingDate,
        'AS' + right('0000' + cast(@BatchKiosk as varchar(4)), 4)

```

```

+ -- as OriginationPoint,
,
-- as Filler
)

5
-- Record Type D
-- Payment Detail
declare DetailCursor cursor for
SELECT
10      'D'
      + -- as RecordType,
      cast(max(BillDetailTransaction.BillAccountNumber) as char(11))
      + -- as CustomerAccountNumber,
      '030'
15      + -- as OfficeCode,
      r i g h t ( ' 0 0 0 0 0 0 0 0 0 ' +
cast(cast(round(((sum(PaymentRecordTransaction.PaymentAmount) -
max(BillDetailTransaction.ServiceAmt)) * 100),0) as int) as varchar(9)),9)
      + -- as AmountPaid,
20      right('0' + cast(datepart(mm,dateadd(day, 1, getdate())) as varchar(2)),2) +
      right('0' + cast(datepart(dd,dateadd(day, 1, getdate())) as varchar(2)),2) +
      cast(right(datepart(yy,dateadd(day, 1, getdate())) as char(2))
      + -- as AgentDate,
      right('0' + cast(datepart(mm,max(BillDetailTransaction.PaybyDate)) as
25      varchar(2)),2) +
      right('0' + cast(datepart(dd,max(BillDetailTransaction.PaybyDate)) as
      varchar(2)),2) +
      cast(right(datepart(yy,max(BillDetailTransaction.PaybyDate)),2) as char(2))
      + -- as DueDate,
30      case max(BillDetailTransaction.CompanyAccountTypeId)
          when '2' then 'G'
          when '9' then 'M'
      end
      + -- as TypeOfBill,
35      ,
      -- as Filler

FROM
      BillDetailTransaction
      JOIN BillPay ON BillDetailTransaction.BillDetailId = BillPay.BillDetailId and
40      BillPay.KioskPCID = BillDetailTransaction.KioskPCID and BillPay.DateStamp =
      BillDetailTransaction.DateTimeofBillScan
      JOIN PaymentRecordTransaction ON BillPay.PaymentRecordId =
      PaymentRecordTransaction.PaymentRecordId and BillPay.KioskPCID =

```

```

PaymentRecordTransaction.KioskPCID and BillPay.DateStamp =
PaymentRecordTransaction.DateTimeofPayment
WHERE
    (BillDetailTransaction.CompanyAccountId = '2' OR
5      BillDetailTransaction.CompanyAccountId = '9') and
      BillDetailTransaction.KioskPCID = @BatchKiosk and
      BillDetailTransaction.QBTed = 10 and
      PaymentRecordTransaction.PaymentAmount > 0
group by BillDetailTransaction.billdetailid
10      having (sum(paymentrecordtransaction.PaymentAmount) -
max(billdetailtransaction.ServiceAmt)) > 0

open DetailCursor
fetch from DetailCursor
15 into @OutputLine

while @@fetch_status = 0
begin
    select @FileRecordCount = @FileRecordCount + 1
20      insert into ##ENTEX_PMT (OutputLine)
        values (@OutputLine)
        fetch next from DetailCursor
        into @OutputLine
end
25

close DetailCursor
deallocate DetailCursor

30      fetch next from BatchCursor
        into @BatchKiosk

end

35      close BatchCursor
        deallocate BatchCursor

-- Record Type T
-- Batch Trailer
40      insert into ##ENTEX_PMT (OutputLine)
        values
        (
            'T'
            + -- as RecordType,

```

```

        right('0' + cast(datepart(mm,dateadd(day, 1, getdate())) as varchar(2)),2) +
        right('0' + cast(datepart(dd,dateadd(day, 1, getdate())) as varchar(2)),2) +
        cast(right(datepart(yy,dateadd(day, 1, getdate())) as char(2))
+ --      as DateTransactionClosed,
5      right('000' + cast(@BatchCount as varchar(3)),3)
+ --      as NumberOfBatches,
      right('00000000' + cast(@FileRecordCount as varchar(8)),8)
+ --      as NumberOfDetailRecords,
      right('0000000000' + cast(cast(round((@FileTotal * 100),0) as int) as varchar(10)),10)
10     + --      as TotalAmountOfPayments,
      '      '
+ --      as AgencyName,
      '      '
--      as Filler
15 )

-- write wire transfer number to table
insert into
      MainSecurity.dbo.WireTransfers
20     (Payee, WireDate, WireAmount, Comment)
values
      ('Entex',   dateadd(day,@WireOffset,getdate()),   @FileTotal,   'Generated   ' +
      convert(varchar,getdate(),101))

25  -- print contents of temp table to disk file
      declare @BCPCommand varchar(200)
      select @BCPCommand = 'bcp "SELECT OutputLine FROM ##ENTEX_PMT order by id"
      queryout "' + @FileName + "' /T /c'
      EXEC master..xp_cmdshell @BCPCommand
30
      select  @BCPCommand = 'move   ' + @FileName + '
      \\ASKVTS01\Transmissions\outgoing\Entex\'
      EXEC master..xp_cmdshell @BCPCommand

35  insert into
      MainSecurity.dbo.JobRC
      (DateTime, ReturnCode, JobTxt, TxtComment)
      values
      (getdate(), 0, 'Entex PMT file', 'File generated successfully.')
40
      drop table ##Entex_PMT

      end -- create proc

```

```
-- Entex NSF report
-- Human-readable report, since Entex has no automated NSF process
```

```
*****
```

```
5  -- RL 01/10/2001 : query written as example
   -- RL 01/19/2001 : store name/number added, with return code
   -- RL 01/25/2001 : send output to text file on ASKVTS01
   -- RL 01/27/2001 : use QBTEd value to choose records to process
   -- RL 01/29/2001 : convert to stored procedure
10  -- RL 03/14/2001 : correct conditions on NSF query
   -- RL 03/29/2001 : add datetime values to join conditions
```

```
use MainKiosk
go
```

```
15  drop proc pr_Entex_NSF
    go
```

```
20  create proc pr_Entex_NSF
    as begin
```

```
   -- overhead for file creation
   create table ##ENTEX_NSF
   (id int identity, OutputLine varchar(200))
25  delete from ##ENTEX_NSF
   declare @FileName varchar(100)
```

```
   select @FileName = 'c:\askHarris_Entex_NSFhuman_' + replace(convert(varchar, getdate(),
30  106), ' ', '-') + '-' + right('0' + cast(datepart(hh,getdate()) as varchar(2)),2) + right('0' +
   cast(datepart(mi,getdate()) as varchar(2)),2) + '.txt'
```

```
   declare @ReturnedAmount float
   declare @ReturnedTotal float
   declare @ReturnedCount int
```

```
35  select  @ReturnedAmount = 0,
           @ReturnedTotal = 0,
           @ReturnedCount = 0
```

```
40  declare @OutputLine varchar(200)
```

```
   insert into ##ENTEX_NSF (OutputLine)
   values ('')
```



```

        join ReturnedChecks rc on prt.PaymentRecordID = rc.PaymentRecordID and
prt.KioskPCID = rc.KioskPCID
        join BillPay bp on prt.PaymentRecordID = bp.PaymentRecordID and prt.KioskPCID
= bp.KioskPCID and prt.DateTimeofPayment = bp.DateStamp
5        join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and bdt.KioskPCID
= bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
        join CustomerIdentificationTransaction cit on cit.CustomerIdentificationID =
prt.CustomerIdentificationID and cit.KioskPCID = prt.KioskPCID
        join ReceiptBill rb on rb.KioskPCID = bp.KioskPCID and rb.BillPayId = bp.BillPayID
10 and rb.DateStamp = bp.DateStamp
        join ReceiptDetailTransaction rdt on rdt.KioskPCID = rb.KioskPCID and
rdt.ReceiptDetailID = rb.ReceiptDetailID and rdt.DateTimeofReceiptIssue = rb.DateStamp
        join MainSecurity.dbo.KioskPCInfo kpc on kpc.KioskPCID = prt.KioskPCID
        join MainSecurity.dbo.KioskInfoMaster kim on kpc.KioskID = kim.KioskInfoID
15 join MainSecurity.dbo.StoreChainProfileMaster scp on kim.StoreChainProfileID =
scp.StoreChainProfileID
        join MainSecurity.dbo.StoreMaster sm on sm.StoreID = kim.StoreID
        join MainSecurity.dbo.AddressMaster am on am.StoreID = kim.StoreID
where
20 bdt.CompanyAccountTypeId in (2,9) and
rc.QBTed = 10
order by
prt.DateTimeofPayment

25 open NSFCursor
fetch from NSFCursor into @OutputLine, @ReturnedAmount

while @@fetch_status = 0
begin
30 select @ReturnedTotal = @ReturnedTotal + @ReturnedAmount
select @ReturnedCount = @ReturnedCount + 1

insert into ##ENTEX_NSF (OutputLine)
values (@OutputLine)
35 fetch next from NSFCursor into @OutputLine, @ReturnedAmount
end

close NSFCursor
deallocate NSFCursor

40 insert into ##ENTEX_NSF (OutputLine)
values (")
insert into ##ENTEX_NSF (OutputLine)

```



```

values ('-----')
insert into ##ENTEX_NSF (OutputLine)
values ('Returned Checks      ' + convert(char(5), @ReturnedCount, 0) +
convert(char(12), cast(@ReturnedTotal as money), 0))
5  insert into ##ENTEX_NSF (OutputLine)
values ('Bank Fees          ' + convert(char(12), cast(@ReturnedCount * 2 as money),
0))
insert into ##ENTEX_NSF (OutputLine)
values ('askHarris Service Charge      ' + + convert(char(12), cast(@ReturnedCount as
10 money), 0))
insert into ##ENTEX_NSF (OutputLine)
values

('=====')
15  insert into ##ENTEX_NSF (OutputLine)
values ('TOTAL                                ' + convert(char(12), cast(@ReturnedTotal +
(@ReturnedCount * 3) as money), 0))
insert into ##ENTEX_NSF (OutputLine)
20 values ('')
insert into ##ENTEX_NSF (OutputLine)
values ('ACH Payment Instructions:')
insert into ##ENTEX_NSF (OutputLine)
values ('Account: askHarris.com')
25 insert into ##ENTEX_NSF (OutputLine)
values ('Bank   : Chase Bank of Texas')
insert into ##ENTEX_NSF (OutputLine)
values ('ABA #   : 111000609')
insert into ##ENTEX_NSF (OutputLine)
30 values ('Account #30802536134')

-- print contents of temp table to disk file
35 declare @BCPCommand varchar(200)
select @BCPCommand = 'bcp "SELECT OutputLine FROM ##ENTEX_NSF order by id"
queryout "' + @FileName + "' /T /c'
EXEC master..xp_cmdshell @BCPCommand

40 select @BCPCommand = 'move ' + @FileName + '
\\ASKVTS01\Transmissions\outgoing\Entex\'
EXEC master..xp_cmdshell @BCPCommand

```

```

insert into
    MainSecurity.dbo.JobRC
    (DateTime, ReturnCode, JobTxt, TxtComment)
values
5     (getdate(), 0, 'Entex NSF file', 'File generated successfully.')

drop table ##Entex_NSF

end -- create proc
10

```

```

-- CHW NSF detail report
-- to be faxed to cashiers
-
*****
5  -- RL 01/17/2001 : file created, just query
   -- RL 01/18/2001 : use cursors, generate totals, print
   -- RL 01/25/2001 : send output to text file on ASKVTS01
   -- RL 01/27/2001 : use QBTed value to choose records to process
   -- RL 01/29/2001 : convert to stored procedure
10 -- RL 03/29/2001 : add datetime values to join conditions

   use MainKiosk
   go

15  drop proc pr_CHW_NSF_report
   go

   create proc pr_CHW_NSF_report
   as begin
20  -- overhead for file creation
   create table ##CHW_FAX
   (id int identity, OutputLine varchar(200))
   delete from ##CHW_FAX
25  declare @FileName varchar(100)

   select @FileName = 'c:\askHarris_CHW_NSFhuman_' + replace(convert(varchar, getdate(),
106), ' ', '-') + '-' + right('0' + cast(datepart(hh, getdate()) as varchar(2)), 2) + right('0' +
   cast(datepart(mi, getdate()) as varchar(2)), 2) + '.txt'
30

   declare @OutputLine varchar(200)
   declare @CheckCount float
   declare @CheckAmount money
35  declare @CheckTotal money
   declare @BankFee money
   declare @BankTotal money

   select @CheckCount = 0
40  select @CheckAmount = 0
   select @CheckTotal = 0
   select @BankFee = 0
   select @BankTotal = 0

```

declare Something cursor for

select

' ' + cast(scp.StoreChainName + ' ' + cast(right(sm.StoreName,3) as char(3)) as char(20))  
+ ' ' + -- as StoreID,

5

bdt.BillAccountNumber

+ ' ' + -- as AccountNbr,

convert(char(12), (prt.PaymentAmount - bdt.ServiceAmt - IsNull(bdt.Donation, 0)), 0)

+ ' ' + -- as PaymentAmount,

' 2.00'

10

+ ' ' + -- as BankFee,

convert(char(12), (2 + prt.PaymentAmount - bdt.ServiceAmt - IsNull(bdt.Donation, 0)),

0)

+ ' ' + -- as Total,

rc.ReturnCode + ' ' + prt.CheckRoutingNumber

15

as Part1,

prt.PaymentAmount - bdt.ServiceAmt - IsNull(bdt.Donation, 0) as CheckAmount,

2 as BankFee

from

BillDetailTransaction bdt

20

JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =  
bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp

JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId  
and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment

JOIN ReturnedChecks rc on rc.KioskPCID = prt.KioskPCID and rc.PaymentRecordID  
= prt.PaymentRecordID

25

join MainSecurity.dbo.KioskPCInfo kpc on kpc.KioskPCID = prt.KioskPCID

join MainSecurity.dbo.KioskInfoMaster kim on kpc.KioskID = kim.KioskInfoID

join MainSecurity.dbo.StoreChainProfileMaster scp on kim.StoreChainProfileID =

scp.StoreChainProfileID

30

join MainSecurity.dbo.StoreMaster sm on sm.StoreID = kim.StoreID

where

bdt.CompanyAccountTypeID = 4 and

rc.QBTed = 10

order by

35

prt.DateTimeofPayment

insert into ##CHW\_FAX (OutputLine)

values ('')

insert into ##CHW\_FAX (OutputLine)

40

values ('askHarris.com')

insert into ##CHW\_FAX (OutputLine)

values ('RETURNED CHECK REPORT')

insert into ##CHW\_FAX (OutputLine)

```

values ('for City of Houston')
insert into ##CHW_FAX (OutputLine)
values ('Generated ' + convert(varchar, getdate()))
insert into ##CHW_FAX (OutputLine)
5 values ('')
insert into ##CHW_FAX (OutputLine)
values ('          Check      Bank      Return  Bank ')
insert into ##CHW_FAX (OutputLine)
values (' Store      Account  Amount  Fee    Total  Code  ABA ')
10 insert into ##CHW_FAX (OutputLine)
values ('-----')

open Something
15 fetch from Something into @OutputLine, @CheckAmount, @BankFee

while @@fetch_status = 0
begin
20     select @CheckCount = @CheckCount + 1
        select @CheckTotal = @CheckTotal + @CheckAmount
        select @BankTotal = @BankTotal + @BankFee

        insert into ##CHW_FAX (OutputLine)
25     values (@OutputLine)
        fetch next from Something into @OutputLine, @CheckAmount, @BankFee
end

close Something
30 deallocate Something

insert into ##CHW_FAX (OutputLine)
values ('')
insert into ##CHW_FAX (OutputLine)
35 values
('=====')
insert into ##CHW_FAX (OutputLine)
values ('')
insert into ##CHW_FAX (OutputLine)
40 values ('Totals: ' + str(@CheckCount, 10) + ' checks      ' + convert(char(12), @CheckTotal)
        + ' ' + convert(char(8), @BankTotal) + ' ' + convert(char(12), (@CheckTotal + @BankTotal)))

```

```

-- print contents of temp table to disk file
declare @BCPCommand varchar(200)
select @BCPCommand = 'bcp "SELECT OutputLine FROM ##CHW_FAX order by id"
queryout "' + @FileName + "' /T /c'
5 EXEC master..xp_cmdshell @BCPCommand

select @BCPCommand = 'move ' + @FileName + '
\\ASKVTS01\Transmissions\outgoing\CHW\'
EXEC master..xp_cmdshell @BCPCommand
10

drop table ##CHW_FAX

insert into
MainSecurity.dbo.JobRC
15 (DateTime, ReturnCode, JobTxt, TxtComment)
values
(getdate(), 0, 'CHW NSF fax', 'File generated successfully.')

end -- create proc
20

```



```

if datename(dw,getdate()) in ('Thursday','Friday')
    select @WireOffset = 4
else
    select @WireOffset = 2
5
declare @OutputPart1 varchar(200)
declare @OutputPart2 varchar(200)
declare @SequenceNumber int

10 declare @PMTBatchTotal float
declare @PMTItemCount int
declare @NSFBatchTotal float
declare @NSFItemCount int
declare @FileTotal float
15 declare @FileItemCount int
declare @PaymentAmount float

select
    @PMTBatchTotal = 0,
20     @PMTItemCount = 0,
    @NSFBatchTotal = 0,
    @NSFItemCount = 0

25 -- Record Type 1
-- File Header
insert into ##CHW_PMT (OutputLine)
values
(
    '1'
30     + -- as RecordType,
    right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
    right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
    right(cast(datepart(yy,getdate()) as varchar(4)),2)
    + -- as FileDate,
35     'HARRIS'
    -- as FileDescriptor
)

-- begin PMT batch
40
-- Record Type 2
-- Batch Header
insert into ##CHW_PMT (OutputLine)

```



```

values
(
    '2'
    + --    as RecordType,
    right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
5    right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
    right(cast(datepart(yy,getdate()) as varchar(4)),2)
    + --    as FileDate,
    '0'
    + --    as BatchPrefix,
10    '77001'
    -- 77001 - 77799 : regular payment batches
    -- 77800 : credit adjustment batch
    -- 77946 : debit adjustment batch
    -- 77945 : NSF batch
15    + --    as BatchNumber,
    'PMT '
    -- ADJCDT : credit batch
    -- ADJDBT : debit batch
    -- NSFADJ : NSF batch
20    --    as FileDescriptor
)

-- Record Type 5
-- Detail
25 declare DetailCursor cursor for
SELECT
    '5'
    + --    as RecordType,
    right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
30    right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
    right(cast(datepart(yy,getdate()) as varchar(4)),2)
    + --    as FileDate,
    '0'
    + --    as BatchPrefix,
35    '77001'
    -- 77001 - 77799 : regular payment batches
    -- 77800 : credit adjustment batch
    -- 77946 : debit adjustment batch
    -- 77945 : NSF batch
40    --    as BatchNumber,
    as Part1,
    right('0000000000' + cast(cast(round(((sum(prt.PaymentAmount) -
max(bdt.ServiceAmt)) * 100),0) as int) as varchar(10)),10)

```

```

+ --    as PaymentAmount,
,
+ --    as Filler1,
right('0000000' + cast(cast(round((IsNull(max(bdt.Donation), 0) * 100),0) as int) as
5  varchar(7)),7)
-- ***** water fund donation amount
+ --    as Filler2,
left(max(bdt.BillAccountNumber), 11)
+ --    as AccountNumber,
10  right(max(bdt.BillAccountNumber), 1)
+ --    as CheckDigit,
right('0' + cast(datepart(mm,max(prt.DateTimeofPayment)) as varchar(2)),2)+
right('0' + cast(datepart(dd,max(prt.DateTimeofPayment)) as varchar(2)),2)
+ --    as PayDate,
15  '1'
-- 1 : credit
-- 2 : debit
+ --    as PayMethod,
CASE count(*)
20  WHEN 1 THEN
CASE max(prt.PaymentTypeID)
WHEN 1 THEN '0' -- cash
WHEN 2 THEN '1' -- check/money order
END
25  ELSE '2' -- combination payment
END
+ --    as PayType,
right('0' + cast(datepart(hh,max(prt.DateTimeofPayment)) as varchar(2)),2)+
right('0' + cast(datepart(mi,max(prt.DateTimeofPayment)) as varchar(2)),2)+
30  right('0' + cast(datepart(ss,max(prt.DateTimeofPayment)) as varchar(2)),2)
--    as PayTime
as Part2,
(sum(prt.PaymentAmount) - max(bdt.ServiceAmt)) as PaymentAmount
FROM
35  BillDetailTransaction bdt
JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId
and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
40  WHERE
bdt.CompanyAccountTypeId = '4' and
bdt.QBTed = 10 and
prt.PaymentAmount > 0

```

```

group by bdt.billdetailid, bdt.kioskpcid
having (sum(prt.PaymentAmount) - max(bdt.ServiceAmt)) > 0
order by
    max(prt.DateTimeofPayment)

```

5

```

open DetailCursor
fetch from DetailCursor
into @OutputPart1, @OutputPart2, @PaymentAmount

```

10 select @SequenceNumber = 0

```

while @@fetch_status = 0
begin

```

```

    select @SequenceNumber = @SequenceNumber + 1
15    select @PMTItemCount = @PMTItemCount + 1
    select @PMTBatchTotal = @PMTBatchTotal + @PaymentAmount
    insert into ##CHW_PMT (OutputLine)

```

```

    values
    (@OutputPart1 + right('0000' + cast(@SequenceNumber as varchar(4)),4) +
20    @OutputPart2)

```

```

    fetch next from DetailCursor
    into @OutputPart1, @OutputPart2, @PaymentAmount

```

25 end

```

close DetailCursor
deallocate DetailCursor

```

30 -- Record type 8  
-- Batch Trailer

```

insert into ##CHW_PMT (OutputLine)
values

```

```

35 (    '8'
    + --    as RecordType,
    right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
    right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
    right(cast(datepart(yy,getdate()) as varchar(4)),2)
40    + --    as FileDate,
    '0'
    + --    as BatchPrefix,
    '77001'

```

```

-- 77001 - 77799 : regular payment batches
-- 77800 : credit adjustment batch
-- 77946 : debit adjustment batch
-- 77945 : NSF batch
5      + --      as BatchNumber,
        '1'
        + --      as PaymentMethod,
        ' '
        + --      as Filler1,
10      right('0000000000' + cast(cast(round((@PMTBatchTotal * 100),0) as int) as
varchar(10)),10)
        + --      as BatchTotal,
        ' '
        + --      as Filler2,
15      right('00000' + cast(@PMTItemCount as varchar(5)),5)
        + --      as ItemCount,
        '001'
        --      as BatchNumber
    )
20      -- begin NSF batch

-- Record Type 2
-- Batch Header
25      insert into ##CHW_PMT (OutputLine)
values
(
    '2'
    + --      as RecordType,
    right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
30      right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
    right(cast(datepart(yy,getdate()) as varchar(4)),2)
    + --      as FileDate,
    '0'
    + --      as BatchPrefix,
35      '77945'
    -- 77001 - 77799 : regular payment batches
    -- 77800 : credit adjustment batch
    -- 77946 : debit adjustment batch
    -- 77945 : NSF batch
40      + --      as BatchNumber,
        'NSFADJ'
        -- ADJCDT : credit batch
        -- ADJDBT : debit batch

```

```

-- NSFADJ : NSF batch
--      as FileDescriptor
)

5  -- Record Type 5
   -- Detail
   declare DetailCursor cursor for
   SELECT
       '5'
10      + --      as RecordType,
         right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
         right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
         right(cast(datepart(yy,getdate()) as varchar(4)),2)
       + --      as FileDate,
15      '0'
       + --      as BatchPrefix,
         '77945'
       -- 77001 - 77799 : regular payment batches
       -- 77800 : credit adjustment batch
20      -- 77946 : debit adjustment batch
       -- 77945 : NSF batch
       --      as BatchNumber,
       as Part1,
       right('0000000000' + cast(cast(round(((prt.PaymentAmount - bdt.ServiceAmt -
25      IsNull(bdt.Donation, 0)) * 100),0) as int) as varchar(10)),10)
       + --      as PaymentAmount,
         ' '
       + --      as Filler1,
       right('00000000' + cast(cast(round((IsNull(bdt.Donation, 0) * 100),0) as int) as
30      varchar(7)),7)
       -- ***** water fund donation amount
       + --      as Filler2,
       left(bdt.BillAccountNumber, 11)
       + --      as AccountNumber,
35      right(bdt.BillAccountNumber, 1)
       + --      as CheckDigit,
       right('0' + cast(datepart(mm,prt.DateTimeofPayment) as varchar(2)),2)+
       right('0' + cast(datepart(dd,prt.DateTimeofPayment) as varchar(2)),2)
       + --      as PayDate,
40      '1'
       -- 1 : credit
       -- 2 : debit
       + --      as PayMethod,

```

```

        '1' -- check/money order
        + --      as PayType,
        right('0' + cast(datepart(hh,prt.DateTimeofPayment) as varchar(2)),2)+
        right('0' + cast(datepart(mi,prt.DateTimeofPayment) as varchar(2)),2)+
5      right('0' + cast(datepart(ss,prt.DateTimeofPayment) as varchar(2)),2)
        --      as PayTime
        as Part2,
        (prt.PaymentAmount - bdt.ServiceAmt - IsNull(bdt.Donation, 0)) as PaymentAmount
FROM
10      BillDetailTransaction bdt
        JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
        bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
        JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId
        and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
15      join ReturnedChecks rc on prt.PaymentRecordID = rc.PaymentRecordID and
        prt.KioskPCID = rc.KioskPCID
WHERE
        bdt.CompanyAccountTypeId = '4' and
        rc.QBTed = 10
20      order by
        prt.DateTimeofPayment

open DetailCursor
fetch from DetailCursor
25      into @OutputPart1, @OutputPart2, @PaymentAmount

select @SequenceNumber = 0

while @@fetch_status = 0
30      begin
        select @SequenceNumber = @SequenceNumber + 1
        select @NSFItemCount = @NSFItemCount + 1
        select @NSFBatchTotal = @NSFBatchTotal + @PaymentAmount
        insert into ##CHW_PMT (OutputLine)
35      values
        (@OutputPart1 + right('0000' + cast(@SequenceNumber as varchar(4)),4) +
        @OutputPart2)

        fetch next from DetailCursor
40      into @OutputPart1, @OutputPart2, @PaymentAmount

end

```

```
close DetailCursor
deallocate DetailCursor
```

```
-- Record type 8
5  -- Batch Trailer
```

```
insert into ##CHW_PMT (OutputLine)
values
```

```
(
10      '8'
      + --      as RecordType,
      right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
      right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
      right(cast(datepart(yy,getdate()) as varchar(4)),2)
      + --      as FileDate,
15      '0'
      + --      as BatchPrefix,
      '77945'
      -- 77001 - 77799 : regular payment batches
      -- 77800 : credit adjustment batch
20      -- 77946 : debit adjustment batch
      -- 77945 : NSF batch
      + --      as BatchNumber,
      '1'
      + --      as PaymentMethod,
25      ' '
      + --      as Filler1,
      right('0000000000' + cast(cast(round((@NSFBatchTotal * 100),0) as int) as
varchar(10)),10)
      + --      as BatchTotal,
30      ' '
      + --      as Filler2,
      right('00000' + cast(@NSFItemCount as varchar(5)),5)
      + --      as ItemCount,
      '002'
35      --      as BatchNumber
)
```

```
-- Record Type 9
-- File Trailer
```

```
40 insert into ##CHW_PMT (OutputLine)
values
(
      '9'
      + --      as RecordType,
```

```

        '
        + -- as Filler1,
        '1'
        + -- as PayMethod,
5        ' '
        + -- as Filler2,
        right('0000000000' + cast(cast(round((((@PMTBatchTotal - @NSFBatchTotal) * 100),0)
as int) as varchar(10)),10)
        + -- as BatchTotal,
10        ' '
        + -- as Filler3,
        right('00000' + cast((@PMTItemCount + @NSFItemCount) as varchar(5)),5)
        + -- as ItemCount,
        '002'
15        -- as BatchCount
    )

    -- write wire transfer number to table
    insert into
20        MainSecurity.dbo.WireTransfers
        (Payee, WireDate, WireAmount, Comment)
    values
        ('City of Houston', dateadd(day,@WireOffset,getdate()), @PMTBatchTotal, 'Generated
        ' + convert(varchar,getdate(),101))
25

    -- print contents of temp table to disk file
    declare @BCPCommand varchar(200)
    select @BCPCommand = 'bcp "SELECT OutputLine FROM ##CHW_PMT order by id"
    queryout "' + @FileName + "' /T /c'
30    EXEC master..xp_cmdshell @BCPCommand

    select @BCPCommand = 'move ' + @FileName + '
    \\ASKVTS01\Transmissions\outgoing\CHW\'
    EXEC master..xp_cmdshell @BCPCommand
35

    delete from ##CHW_PMT

    insert into ##CHW_PMT (OutputLine)
40    values
        ('<html><head><title>File Transmission Report</title></head><body
        bgcolor="#FFFFFF"><div align="center">')
    insert into ##CHW_PMT (OutputLine)

```



```

values
('(<font size="2">Please fax this page to (713) 371-1037</font>')
insert into ##CHW_PMT (OutputLine)
values
5 ('(<H2>askHarris.com<br>PAYMENT FILE TRANSMISSION REPORT</br>for City of
Houston Water</h2>')
insert into ##CHW_PMT (OutputLine)
values
('(<b>generated ' + convert(varchar, getdate()) + '</b><hr>')
10 insert into ##CHW_PMT (OutputLine)
values
('(<font size="+1"><p>' + cast((@PMTItemCount + @NSFItemCount) as varchar(5)) + '
records</p>')
insert into ##CHW_PMT (OutputLine)
15 values
('(<p>' + cast(cast((@PMTBatchTotal + @NSFBatchTotal) as money) as varchar(20)) + ' total
amount</p></font>')
insert into ##CHW_PMT (OutputLine)
values
20 ('(<p>&nbsp;</p><p>If there are any problems with this transmission, please call (713)
935-3605.</p>')
insert into ##CHW_PMT (OutputLine)
values
('(</div></body></html>')
25
select @FileName = replace(@FileName, '.rpt', '.alert.htm')
select @BCPCommand = 'bcp "SELECT OutputLine FROM ##CHW_PMT order by id"
queryout "' + @FileName + "' /T /c'
EXEC master..xp_cmdshell @BCPCommand
30
select @BCPCommand = 'move ' + @FileName + '
\\ASKVTS01\Transmissions\outgoing\CHW\
EXEC master..xp_cmdshell @BCPCommand

35 insert into
MainSecurity.dbo.JobRC
(DateTime, ReturnCode, JobTxt, TxtComment)
values
(getdate(), 0, 'CHW PMT/NSF file', 'File generated successfully.')
40
drop table ##CHW_PMT

end -- create proc

```

-- CHW PMT detail report

-- to be faxed to cashiers

-

\*\*\*\*\*

5 -- RL 01/17/2001 : file created, just query

-- RL 01/18/2001 : use cursors, generate totals, write to output file

-- RL 01/18/2001 : change query for payments, add totals

-- RL 01/25/2001 : send output to text file on ASKVTS01

-- RL 01/27/2001 : use QBTed value to choose records to process

10 -- RL 01/29/2001 : convert to stored procedure

-- RL 02/03/2001 : correct for mixed payments (one payment, one service charge)

-- RL 02/09/2001 : prevent reporting of zero payments

-- RL 02/10/2001 : prevent reporting of amounts <=0 (\$1 payment - \$1 svc chg)

-- RL 03/29/2001 : add datetime values to join conditions

15

use MainKiosk

go

drop proc pr\_CHW\_PMT\_report

20

go

create proc pr\_CHW\_PMT\_report

as begin

25

-- overhead for file creation

create table ##CHW\_FAX

(id int identity, OutputLine varchar(200))

delete from ##CHW\_FAX

declare @FileName varchar(100)

30

select @FileName = 'c:\askHarris\_CHW\_PMT\human\_' + replace(convert(varchar, getdate(), 106), ' ', ',') + '-' + right('0' + cast(datepart(hh, getdate()) as varchar(2)), 2) + right('0' + cast(datepart(mi, getdate()) as varchar(2)), 2) + '.txt'

35

declare @OutputLine varchar(200)

declare @QueryCount int

declare @QueryTotal money

40

declare @SummaryCount int

declare @SummaryTotal money

select @SummaryCount = 0

select @SummaryTotal = 0

declare Something cursor for  
select

```
5      ' ' +
      right('00' + cast(datepart(mm, max(prt.DateTimeofPayment)) as varchar(2)),2) + '/' +
right('00' + cast(datepart(dd, max(prt.DateTimeofPayment)) as varchar(2)),2)
      + ' ' +
      right('00' + cast(datepart(hh, max(prt.DateTimeofPayment)) as varchar(2)),2) + ':' +
10    right('00' + cast(datepart(mi, max(prt.DateTimeofPayment)) as varchar(2)),2)
      + ' ' +
      max(bdt.BillAccountNumber)
      + ' ' +
      convert(char(15), (sum(prt.PaymentAmount) - max(bdt.ServiceAmt) -
15    IsNull(max(bdt.Donation), 0)), 0)
      + ' ' +
      case count(*)
      when 1 then cast(max(pt.PaymentTypeName) as char(8))
      else 'Mixed '
20    end
      + ' ' +
      max(rdt.ReceiptNumber)
from
      BillDetailTransaction bdt
25    JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
      bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
      JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId
      and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
      JOIN PaymentType pt on pt.PaymentTypeId = prt.PaymentTypeId
30    JOIN ReceiptBill rb on rb.BillPayID = bp.BillPayID and rb.KioskPCID =
      bp.KioskPCID and rb.DateStamp = bp.DateStamp
      JOIN ReceiptDetailTransaction rdt on rdt.ReceiptDetailID = rb.ReceiptDetailID and
      rb.KioskPCID = rdt.KioskPCID and rb.DateStamp = rdt.DateTimeofReceiptIssue
where
35    bdt.CompanyAccountTypeID = 4 and
      bdt.QBTed = 10 and
      prt.PaymentAmount > 0
group by bdt.billdetailid, bdt.kioskpcid
having (sum(prt.PaymentAmount) - max(bdt.ServiceAmt)) > 0
40 order by
      max(prt.DateTimeofPayment)

insert into ##CHW_FAX (OutputLine)
```

```

values ("
insert into ##CHW_FAX (OutputLine)
values ('askHarris.com')
insert into ##CHW_FAX (OutputLine)
5 values ('PAYMENT FILE REPORT')
insert into ##CHW_FAX (OutputLine)
values ('for City of Houston')
insert into ##CHW_FAX (OutputLine)
values ('Generated ' + convert(varchar, getdate()))
10 insert into ##CHW_FAX (OutputLine)
values ("
insert into ##CHW_FAX (OutputLine)
values (' Date Time Account Amount Type Receipt #')
insert into ##CHW_FAX (OutputLine)
15 values ('-----')

open Something
fetch from Something into @OutputLine
20
while @@fetch_status = 0
begin
insert into ##CHW_FAX (OutputLine)
values (@OutputLine)
25 fetch next from Something into @OutputLine
end

close Something
deallocate Something
30
insert into ##CHW_FAX (OutputLine)
values ("
insert into ##CHW_FAX (OutputLine)
values
35 ('=====')
insert into ##CHW_FAX (OutputLine)
values ("

40 declare Something cursor for
select
cast((PaymentType + ' Transactions') as char(20))
+ ' ' +

```

```

        cast(isnull(count(*),0) as char(10))
        + ' ' +
        convert(char(15), isnull(sum(PaymentAmount),0),0),
        isnull(count(*),0), sum(PaymentAmount)
5    from
    (select
        case count(*)
            when 1 then max(pt.PaymentTypeName)
            else 'Mixed'
10       end as PaymentType,
        (sum(prt.PaymentAmount) - max(bdt.ServiceAmt) - IsNull(max(bdt.Donation), 0)) as
        PaymentAmount
    from
        BillDetailTransaction bdt
15     JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
        bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
        JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId
        and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
        JOIN PaymentType pt on pt.PaymentTypeID = prt.PaymentTypeid
20     JOIN ReceiptBill rb on rb.BillPayID = bp.BillPayID and rb.KioskPCID =
        bp.KioskPCID and rb.DateStamp = bp.DateStamp
        JOIN ReceiptDetailTransaction rdt on rdt.ReceiptDetailID = rb.ReceiptDetailID and
        rb.KioskPCID = rdt.KioskPCID and rb.DateStamp = rdt.DateTimeofReceiptIssue
    where
25     bdt.CompanyAccountTypeID = 4 and
        bdt.QBTed = 10 and
        prt.PaymentAmount > 0
    group by bdt.billdetailid, bdt.kioskpcid) as foo
    group by PaymentType
30
    open Something
    fetch from Something into @OutputLine, @QueryCount, @QueryTotal

    while @@fetch_status = 0
35    begin
        select @SummaryCount = @SummaryCount + @QueryCount
        select @SummaryTotal = @SummaryTotal + @QueryTotal

        insert into ##CHW_FAX (OutputLine)
40       values (@OutputLine)
        fetch next from Something into @OutputLine, @QueryCount, @QueryTotal
    end

```

```
close Something
deallocate Something
```

```
insert into ##CHW_FAX (OutputLine)
```

```
5 values ('Total Transactions ' + cast(@SummaryCount as char(10)) + ' ' + convert(char(15),
@SummaryTotal))
```

```
-- print contents of temp table to disk file
```

```
10 declare @BCPCommand varchar(200)
```

```
select @BCPCommand = 'bcp "SELECT OutputLine FROM ##CHW_FAX order by id"
queryout "' + @FileName + "' /T /c'
```

```
EXEC master..xp_cmdshell @BCPCommand
```

```
15 select @BCPCommand = 'move ' + @FileName + '
\\ASKVTS01\Transmissions\outgoing\CHW'
```

```
EXEC master..xp_cmdshell @BCPCommand
```

```
20 insert into
```

```
    MainSecurity.dbo.JobRC
```

```
    (DateTime, ReturnCode, JobTxt, TxtComment)
```

```
values
```

```
    (getdate(), 0, 'CHW PMT fax', 'File generated successfully.')
```

```
25
```

```
drop table ##CHW_FAX
```

```
end -- create proc
```

```
30
```

-- City of Houston Water MEMO file

-

\*\*\*\*\*

-- AG & RL 11/25/2000 : insert detail records to CHW\_PMT table

5 -- RL 11/29/2000 : no insertion, wrap in header/footer

-- RL 12/12/2000 : use cursors/print, generate sequence number, account for service charge and water fund donation

-- RL 01/16/2001 : database changes (donation field)

-- RL 01/16/2001 : QB Ted value of 1 for first day of PMT testing, 2 for second, 3 for third.

10 -- RL 01/16/2001 : file generates follow-on NSF batch, based on the handmade one from earlier test

-- RL 01/25/2001 : send output to test file on ASKVTS01, create file to be faxed to CHW operations

-- RL 01/25/2001 : this file made from PMT file

15 -- RL 01/27/2001 : use QB Ted value to choose records to process

-- RL 01/29/2001 : convert to stored procedure

-- RL 02/03/2001 : correct for mixed payments (one payment, one service charge)

-- RL 02/09/2001 : prevent reporting of zero payments

-- RL 02/10/2001 : prevent reporting of amounts <=0 (\$1 payment - \$1 svc chg)

20 -- RL 03/29/2001 : add datetime values to join conditions

use MainKiosk

go

25 drop proc pr\_CHW\_memo

go

create proc pr\_CHW\_memo

as begin

30

-- overhead for file creation

create table ##CHW\_PMT

(id int identity, OutputLine varchar(200))

delete from ##CHW\_PMT

35 declare @FileName varchar(100)

select @FileName = 'c:\askHarris\_CHW\_MEMO\_' + replace(convert(varchar, getdate(), 106),  
' ', ',') + '-' + right('0' + cast(datepart(hh, getdate()) as varchar(2)), 2) + right('0' +  
40 cast(datepart(mi, getdate()) as varchar(2)), 2) + '.rpt'

declare @OutputPart1 varchar(200)

declare @OutputPart2 varchar(200)

```
declare @SequenceNumber int
```

```
declare @PMTBatchTotal float
```

```
declare @PMTItemCount int
```

```
5 declare @FileTotal float
```

```
declare @FileItemCount int
```

```
declare @PaymentAmount float
```

```
select
```

```
10      @PMTBatchTotal = 0,
      @PMTItemCount = 0
```

```
-- Record Type 1
```

```
15 -- File Header
```

```
insert into ##CHW_PMT (OutputLine)
```

```
values
```

```
(      '1'
```

```
      + --      as RecordType,
```

```
20      right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
```

```
      right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
```

```
      right(cast(datepart(yy,getdate()) as varchar(4)),2)
```

```
      + --      as FileDate,
```

```
      'HARRIS'
```

```
25      --      as FileDescriptor
```

```
)
```

```
-- begin PMT batch
```

```
30 -- Record Type 2
```

```
-- Batch Header
```

```
insert into ##CHW_PMT (OutputLine)
```

```
values
```

```
(      '2'
```

```
      + --      as RecordType,
```

```
      right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
```

```
      right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
```

```
      right(cast(datepart(yy,getdate()) as varchar(4)),2)
```

```
      + --      as FileDate,
```

```
40      '0'
```

```
      + --      as BatchPrefix,
```

```
      '77001'
```

```
      -- 77001 - 77799 : regular payment batches
```



```

-- 77800 : credit adjustment batch
-- 77946 : debit adjustment batch
-- 77945 : NSF batch
+ --      as BatchNumber,
5  'PMT '
-- ADJCDT : credit batch
-- ADJDBT : debit batch
-- NSFADJ : NSF batch
--      as FileDescriptor
10 )

-- Record Type 5
-- Detail
declare DetailCursor cursor for
15 SELECT
    '5'
    + --      as RecordType,
    right('0' + cast(datepart(mm,getdate()) as varchar(2)),2)+
    right('0' + cast(datepart(dd,getdate()) as varchar(2)),2)+
20    right(cast(datepart(yy,getdate()) as varchar(4)),2)
    + --      as FileDate,
    '0'
    + --      as BatchPrefix,
    '77001'
25    -- 77001 - 77799 : regular payment batches
    -- 77800 : credit adjustment batch
    -- 77946 : debit adjustment batch
    -- 77945 : NSF batch
    --      as BatchNumber,
30    as Part1,
    right('0000000000' + cast(cast(round(((sum(prt.PaymentAmount) -
max(bdt.ServiceAmt) - IsNull(max(bdt.Donation), 0)) * 100),0) as int) as varchar(10)),10)
    + --      as PaymentAmount,
    ' '
35    + --      as Filler1,
    right('0000000' + cast(cast(round((IsNull(max(bdt.Donation), 0) * 100),0) as int) as
varchar(7)),7)
    -- ***** water fund donation amount
    + --      as Filler2,
40    left(max(bdt.BillAccountNumber), 11)
    + --      as AccountNumber,
    right(max(bdt.BillAccountNumber), 1)
    + --      as CheckDigit,

```

```

right('0' + cast(datepart(mm,max(prt.DateTimeofPayment)) as varchar(2)),2)+
right('0' + cast(datepart(dd,max(prt.DateTimeofPayment)) as varchar(2)),2)
+ --    as PayDate,
'1'
5      -- 1 : credit
      -- 2 : debit
      + --    as PayMethod,
      CASE count(*)
10     WHEN 1 THEN
          CASE max(prt.PaymentTypeID)
            WHEN 1 THEN '0' -- cash
            WHEN 2 THEN '1' -- check/money order
          END
        ELSE '2' -- combination payment
15     END
      + --    as PayType,
      right('0' + cast(datepart(hh,max(prt.DateTimeofPayment)) as varchar(2)),2)+
      right('0' + cast(datepart(mi,max(prt.DateTimeofPayment)) as varchar(2)),2)+
      right('0' + cast(datepart(ss,max(prt.DateTimeofPayment)) as varchar(2)),2)
20     --    as PayTime
      as Part2,
      (sum(prt.PaymentAmount) - max(bdt.ServiceAmt) - IsNull(max(bdt.Donation), 0)) as
PaymentAmount
FROM
25     BillDetailTransaction bdt
      JOIN BillPay bp ON bdt.BillDetailId = bp.BillDetailId and bdt.KioskPCID =
bp.KioskPCID and bdt.DateTimeofBillScan = bp.DateStamp
      JOIN PaymentRecordTransaction prt ON bp.PaymentRecordId = prt.PaymentRecordId
and bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
30     WHERE
          bdt.CompanyAccountTypeId = '4' and
          bdt.QBTed = 10 and
          prt.PaymentAmount > 0
      group by bdt.billdetailid, bdt.kioskpcid
35     having (sum(prt.PaymentAmount) - max(bdt.ServiceAmt)) > 0
      order by
          max(prt.DateTimeofPayment)

      open DetailCursor
40     fetch from DetailCursor
      into @OutputPart1, @OutputPart2, @PaymentAmount

      select @SequenceNumber = 0

```



```

        + --      as BatchTotal,
        ,
        + --      as Filler2,
        right('00000' + cast(@PMTItemCount as varchar(5)),5)
5      + --      as ItemCount,
        '001'
        --      as BatchNumber
    )

10      -- Record Type 9
      -- File Trailer
      insert into ##CHW_PMT (OutputLine)
      values
15      (
          '9'
          + --      as RecordType,
          ,
          + --      as Filler1,
          '1'
20      + --      as PayMethod,
          ,
          + --      as Filler2,
          right('0000000000' + cast(cast(round(((@PMTBatchTotal * 100),0) as int) as
          varchar(10)),10)
25      + --      as BatchTotal,
          ,
          + --      as Filler3,
          right('00000' + cast(@PMTItemCount as varchar(5)),5)
          + --      as ItemCount,
30      '002'
          --      as BatchCount
      )

      -- print contents of temp table to disk file
35      declare @BCPCommand varchar(200)
      select @BCPCommand = 'bcp "SELECT OutputLine FROM ##CHW_PMT order by id"
      queryout "' + @FileName + "' /T /c'
      EXEC master..xp_cmdshell @BCPCommand

40      select @BCPCommand = 'move ' + @FileName + '
      \\ASKVTS01\Transmissions\outgoing\CHW\
      EXEC master..xp_cmdshell @BCPCommand

```

delete from ##CHW\_PMT

insert into ##CHW\_PMT (OutputLine)  
values

5 ('<html><head><title>File Transmission Report</title></head><body  
bgcolor="#FFFFFF"><div align="center">')

insert into ##CHW\_PMT (OutputLine)  
values

10 ('<font size="2">Please fax this page to (713) 371-1037</font>')

insert into ##CHW\_PMT (OutputLine)  
values  
(('<H2>askHarris.com<br>MEMO FILE TRANSMISSION REPORT</br>for City of Houston  
Water</h2>')

15 insert into ##CHW\_PMT (OutputLine)  
values

('<b>generated ' + convert(varchar, getdate()) + '</b><hr>')

insert into ##CHW\_PMT (OutputLine)  
values

20 ('<font size="+1"><p>' + cast(@PMTItemCount as varchar(5)) + ' records</p>')

insert into ##CHW\_PMT (OutputLine)  
values  
(('<p>' + cast(cast(@PMTBatchTotal as money) as varchar(20)) + ' total amount</p></font>')

25 insert into ##CHW\_PMT (OutputLine)  
values  
(('<p>&nbsp;</p><p>If there are any problems with this transmission, please call (713)  
935-3605.</p>')

insert into ##CHW\_PMT (OutputLine)  
values

30 ('</div></body></html>')

select @FileName = replace(@FileName, '.rpt', '.alert.htm')

select @BCPCommand = 'bcp "SELECT OutputLine FROM ##CHW\_PMT order by id"  
queryout "' + @FileName + '" /T /c'

EXEC master..xp\_cmdshell @BCPCommand

35 select @BCPCommand = 'move ' + @FileName + '  
\\ASKVTS01\Transmissions\outgoing\CHW'  
EXEC master..xp\_cmdshell @BCPCommand

40 insert into  
MainSecurity.dbo.JobRC  
(DateTime, ReturnCode, JobTxt, TxtComment)  
values

```
(getdate(), 0, 'CHW MEMO file', 'File generated successfully.')
```

```
drop table ##CHW_PMT
```

```
5 end -- create proc
```

For more information

```

-- Chase ACH file generation
-
*****
5  -- RL 12/01/2000 : created from file using unions
   -- RL 12/19/2000 : start on multi-batch file
   --          separate batches for SWB and other checks
   -- RL 12/20/2000 : more multi-batching
   --          cash batches for SWB and other, service charge payouts, cash voucher payouts
10  -- RL 01/26/2001 : match customer with kioskpaid, add isnull() in case of zero-length recordsets
   -- RL 01/27/2001 : do not produce headers or trailers for empty batches
   -- RL 01/27/2001 : use QBTEd value to choose records to process
   -- RL 01/29/2001 : convert to stored procedure
   -- RL 02/05/2001 : add SWB CTX batch at bottom of file
   -- RL 02/09/2001 : allow for zero service charge
15  -- RL 02/09/2001 : set minimum date for customer identification records
   -- RL 02/10/2001 : group SWB payments to prevent double-subtraction of service charge
   -- RL 02/10/2001 : prevent reporting of amounts <=0 to SWB
   -- RL 02/12/2001 : add code to generate proper one-day and two-day dates (weekdays only)
   -- RL 02/15/2001 : correct for CustomerIdentificationTransaction before 02/06 19:30
20  -- RL 02/20/2001 : change 5-record company name from 'askHarris.com' to 'askHarr Bill Pay'
   on check batches
   -- RL 02/25/2001 : clean up queries using JOIN, add ORDER BY on checks
   -- RL 02/25/2001 : record batch totals and askHarris service charge amount to WireTransfers
   table
25  -- RL 03/01/2001 : change float types to money types (for proper string-format display)
   -- RL 03/12/2001 : change int to bigint, to allow for large batch trailer amounts
   -- RL 03/26/2001 : change 'cast as varchar(n)' to 'cast as varchar' to avoid arithmetic overflow
   -- RL 03/27/2001 : process all checks as one-day ACH
   -- RL 03/29/2001 : add datetime fields to join conditions
30  -- RL 03/29/2001 : change WireTransfer dates on ACH items to match posting date
   -- *****

use MainKiosk
go
35

drop proc pr_Chase_ACH
go

create proc pr_Chase_ACH
40 as begin

create table ##SWB_EDI (id int identity, OutputLine char(80))
delete from ##SWB_EDI

```







```

as char(2)) +
    right('0' + cast(datepart(mm,dateadd(day, @OneDayOffset, getdate())) as varchar),2) +
    right('0' + cast(datepart(dd,dateadd(day, @OneDayOffset, getdate())) as varchar),2)

5  -- Effective date for two-day ACH batches
    declare @TwoDayDate char(6)
    select @TwoDayDate = cast(right( datepart(yy,dateadd(day, @TwoDayOffset, getdate())) ,2)
as char(2)) +
        right('0' + cast(datepart(mm,dateadd(day, @TwoDayOffset, getdate())) as varchar),2)
10 +
        right('0' + cast(datepart(dd,dateadd(day, @TwoDayOffset, getdate())) as varchar),2)

    -- File transmission date and time
    declare @FileXmitDate char(6)
15 declare @FileXmitTime char(4)
    select @FileXmitDate = cast(right(datepart(yy,getdate()),2) as char(2)) +
        right('0' + cast(datepart(mm,getdate()) as varchar),2) +
        right('0' + cast(datepart(dd,getdate()) as varchar),2)
    select @FileXmitTime = right('0' + cast(datepart(hh,getdate()) as varchar),2)+
20 right('0' + cast(datepart(mi,getdate()) as varchar),2)

    -- record type 1
    -- File Header
25 insert into ##Chase_ACH (OutputLine)
    values
    (
        '1'
            + -- as RecordType,
        '01'
            + -- as PriorityCode,
30 '021000021'
            + -- as ImmediateDestination,
        '1' + @ChaseCompanyID
            + -- as ImmediateOrigin,
35 @FileXmitDate
            + -- as TransmissionDate,
        @FileXmitTime
            + -- as TransmissionTime,
        'A'
40 + -- as FileIDModifier,
        '094'
            + -- as RecordSize,
        '10'
    )

```

```

        + -- as BlockingFactor,
    '1'
        + -- as FormatCode,
    'CHASE MANHATTAN BANK '
5      + -- as ImmediateDestinationName,
    'askHarris.com '
        + -- as ImmediateOriginName,
    ' '
        -- as ReferenceCode
10    )

-- First Batch: SWB checks

15    -- set up cursor for detail records
    -- PaymentType = 2 for checks
    -- CompanyAccountTypeID = 3 for SWB
    set @EntryDetail = cursor for
    select
20        '6'
            + -- as RecordType,
        '27'
            + -- as TransactionCode,
        CheckRoutingNumber
25        + -- as ReceivingABA,
        cast(CheckingAccountNumber as char(17))
            + -- as DFIAccountNumber,
        right('0000000000' + cast(cast(round((PaymentAmount * 100),0) as int) as varchar),10)
            -- as DollarAmount,
30        as Part1,
        cast(replace(Name, '$', ' ') as char(22))
            + -- as DestinationName
        '1'
            + -- as MediaDetermination,
35        ''
            + -- as DiscretionaryDate,
        '0'
            + -- as AddendaRecordIndicator,
        substring(@ChaseCompanyID, 1, 8)
40        -- as OriginatingDFIID,
        as Part2

    from
        PaymentRecordTransaction prt

```

```

        join BillPay bp on bp.PaymentRecordID = prt.PaymentRecordid and bp.KioskPCID =
prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
        join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and bp.KioskPCID
= bdt.KioskPCID and bp.DateStamp = bdt.DateTimeofBillScan
5      join CustomerIdentificationTransaction cit on prt.CustomerIdentificationID =
cit.CustomerIdentificationID and cit.CheckingAccountNumber1 = prt.CheckingAccountNumber
where
        prt.PaymentTypeID = 2 and
        prt.QBTed = 10 and
10      bdt.CompanyAccountTypeID = 3
order by prt.DateTimeofPayment

-- open cursor and fetch for detail record
open @EntryDetail
15
fetch next from @EntryDetail
into @OutputPart1, @OutputPart2

-- only write batch if there is a detail record set
20 if @@fetch_status = 0
begin
        select @BatchCount = @BatchCount + 1

        -- record type 5
        -- Batch Header
        insert into ##Chase_ACH (OutputLine)
        values
        (
                '5'
                + -- as RecordType,
30      '200'
                + -- as ServiceClassCode,
                'askHarr Bill Pay'
                + -- as CompanyName,
                ,
35      + -- as CompanyDiscretionaryData,
                '2' + @ChaseCompanyID
                + -- as CompanyID,
                'PPD'
                + -- as StandardEntryClassCode,
40      'SWB CHECKS'
                + -- as CompanyEntryDescription,
                @FileXmitDate
                + -- as CompanyDescriptiveDate,

```

```

        @OneDayDate
        + -- as EffectiveEntryDate,
        ,
        + -- as Settlement,
5      '1'
        + -- as OriginatorStatusCode,
        substring(@ChaseCompanyID, 1, 8)
        + -- as OriginatingDFIID,
        right('0000000' + cast(@BatchCount as varchar),7)
10      -- as BatchNumber
    )

    -- record type 6
    -- Entry Detail
15    -- records fetched from cursor above
    while @@fetch_status = 0
    begin
        select @TraceNumber = @TraceNumber + 1
        insert into ##Chase_ACH (OutputLine)
20      values (@OutputPart1 + right('0000000000000000' + cast(@TraceNumber as
varchar),15) + @OutputPart2 + right('0000000' + cast(@TraceNumber as varchar),7))
        fetch next from @EntryDetail
        into @OutputPart1, @OutputPart2
    end
25

    -- record type 8
    -- Batch Control
    -- Figure totals for Batch Control Record (type 8)
    select
30      @BatchEntryCount = isnull(count(*),0),
      @BatchDebitAmount = isnull(sum(PaymentAmount),0),
      @BatchEntryHash = isnull(sum(cast(left(CheckRoutingNumber,8) as bigint)),0)
    from
        PaymentRecordTransaction prt
35      join BillPay bp on bp.PaymentRecordID = prt.PaymentRecordid and
        bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
        join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and
        bp.KioskPCID = bdt.KioskPCID and bp.DateStamp = bdt.DateTimeofBillScan
        join CustomerIdentificationTransaction cit on prt.CustomerIdentificationID =
40      cit.CustomerIdentificationID and cit.CheckingAccountNumber1 = prt.CheckingAccountNumber
    where
        prt.PaymentTypeID = 2 and
        prt.QBTed = 10 and

```

bdt.CompanyAccountTypeID = 3

```
insert into ##Chase_ACH (OutputLine)
5 values
  (      '8'
        + -- as RecordType,
        '200'
        + -- as ServiceClassCode,
10      right('000000' + cast(@BatchEntryCount as varchar),6)
        + -- as EntryCount,
        right('000000000000' + cast(@BatchEntryHash as varchar),10)
        + -- as EntryHash,
        right('00000000000000' + cast(cast(round(((@BatchDebitAmount * 100),0) as
15      bigint) as varchar),12)
        + -- as TotalDebitAmount,
        '00000000000000'
        + -- as TotalCreditAmount,
        '2' + @ChaseCompanyID
        + -- as CompanyID,
20      ,
        + -- as Reserved1,
        ,
        + -- as Reserved2,
        substring(@ChaseCompanyID, 1, 8)
        + -- as OriginatingDFI,
25      right('0000000' + cast(@BatchCount as varchar),7)
        -- as BatchNumber
    )
30
    -- write batch total number to wire transfer table
    insert into
        MainSecurity.dbo.WireTransfers
        (Payee, WireDate, WireAmount, Comment)
35      values
        ('ACH: SWB Checks', dateadd(day,@OneDayOffset,getdate()),
        @BatchDebitAmount, 'Generated ' + convert(varchar,getdate(),101))

    select @FileEntryCount = @FileEntryCount + @BatchEntryCount
40      select @FileEntryHash = @FileEntryHash + @BatchEntryHash
      select @FileDebitAmount = @FileDebitAmount + @BatchDebitAmount

end
```

close @EntryDetail

-- Second Batch: non-SWB checks

5 -- Figure totals for Batch Control Record (type 8)

-- PaymentType = 2 for checks

set @EntryDetail = cursor for  
select

'6'

10 + -- as RecordType,

'27'

+ -- as TransactionCode,

CheckRoutingNumber

+ -- as ReceivingABA,

15 cast(CheckingAccountNumber as char(17))

+ -- as DFIAccountNumber,

right('0000000000' + cast(cast(round((PaymentAmount \* 100),0) as int) as varchar),10)

-- as DollarAmount,

as Part1,

20 cast(replace(Name, '\$', ' ') as char(22))

+ -- as DestinationName

'1'

+ -- as MediaDetermination,

''

25 + -- as DiscretionaryDate,

'0'

+ -- as AddendaRecordIndicator,

substring(@ChaseCompanyID, 1, 8)

-- as OriginatingDFIID,

30 as Part2

from

PaymentRecordTransaction prt

join BillPay bp on bp.PaymentRecordID = prt.PaymentRecordid and bp.KioskPCID =  
prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment

35 join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and bp.KioskPCID  
= bdt.KioskPCID and bp.DateStamp = bdt.DateTimeofBillScan

join CustomerIdentificationTransaction cit on prt.CustomerIdentificationID =  
cit.CustomerIdentificationID and cit.CheckingAccountNumber1 = prt.CheckingAccountNumber

where

40 prt.PaymentTypeID = 2 and

prt.QBTed = 10 and

bdt.CompanyAccountTypeID <> 3

order by prt.DateTimeofPayment

```

open @EntryDetail

fetch next from @EntryDetail
5 into @OutputPart1, @OutputPart2

if @@fetch_status = 0
begin
    select @BatchCount = @BatchCount + 1
10
    select
        @BatchEntryCount = isnull(count(*),0),
        @BatchDebitAmount = isnull(sum(PaymentAmount),0),
        @BatchEntryHash = isnull(sum(cast(left(CheckRoutingNumber,8) as bigint)),0)
15 from
        PaymentRecordTransaction prt
        join BillPay bp on bp.PaymentRecordID = prt.PaymentRecordid and
        bp.KioskPCID = prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
        join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and
20 bp.KioskPCID = bdt.KioskPCID and bp.DateStamp = bdt.DateTimeofBillScan
        join CustomerIdentificationTransaction cit on prt.CustomerIdentificationID =
        cit.CustomerIdentificationID and cit.CheckingAccountNumber1 = prt.CheckingAccountNumber
        where
            prt.PaymentTypeID = 2 and
25 prt.QBTed = 10 and
            bdt.CompanyAccountTypeID <> 3

    -- record type 5
    -- Batch Header
30 insert into ##Chase_ACH (OutputLine)
    values
    (
        '5'
            + -- as RecordType,
        '200'
            + -- as ServiceClassCode,
35 'askHarr Bill Pay'
            + -- as CompanyName,
        ,
            + -- as CompanyDiscretionaryData,
40 '3' + @ChaseCompanyID
            + -- as CompanyID,
        'PPD'
            + -- as StandardEntryClassCode,

```



```

'UTILITY CK'
    + -- as CompanyEntryDescription,
    @FileXmitDate
    + -- as CompanyDescriptiveDate,
5    @OneDayDate
    + -- as EffectiveEntryDate,
    ,
    + -- as Settlement,
    '1'
10    + -- as OriginatorStatusCode,
    substring(@ChaseCompanyID, 1, 8)
    + -- as OriginatingDFIID,
    right('0000000' + cast(@BatchCount as varchar),7)
    -- as BatchNumber
15    )

-- record type 6
-- Entry Detail
while @@fetch_status = 0
20    begin
        select @TraceNumber = @TraceNumber + 1
        insert into ##Chase_ACH (OutputLine)
        values (@@OutputPart1 + right('0000000000000000' + cast(@TraceNumber as
varchar),15) + @@OutputPart2 + right('0000000' + cast(@TraceNumber as varchar),7))
25        fetch next from @EntryDetail
        into @OutputPart1, @OutputPart2
    end

30    -- record type 8
    -- Batch Control
    insert into ##Chase_ACH (OutputLine)
    values
    (
35        '8'
        + -- as RecordType,
        '200'
        + -- as ServiceClassCode,
        right('000000' + cast(@BatchEntryCount as varchar),6)
        + -- as EntryCount,
40        right('00000000000' + cast(@BatchEntryHash as varchar),10)
        + -- as EntryHash,
        right('0000000000000' + cast(cast(round((@BatchDebitAmount * 100),0) as
bigint) as varchar),12)

```

```

        + -- as TotalDebitAmount,
        '000000000000'
        + -- as TotalCreditAmount,
        '3' + @ChaseCompanyID
5         + -- as CompanyID,
        ,
        + -- as Reserved1,
        ,
        + -- as Reserved2,
10      substring(@ChaseCompanyID, 1, 8)
        + -- as OriginatingDFI,
        right('0000000' + cast(@BatchCount as varchar),7)
        -- as BatchNumber
    )
15
    -- write batch total number to wire transfer table
    insert into
        MainSecurity.dbo.WireTransfers
        (Payee, WireDate, WireAmount, Comment)
20    values
        ('ACH: Utility Checks', dateadd(day,@OneDayOffset,getdate()),
        @BatchDebitAmount, 'Generated ' + convert(varchar,getdate(),101))

        select @FileEntryCount = @FileEntryCount + @BatchEntryCount
25      select @FileEntryHash = @FileEntryHash + @BatchEntryHash
        select @FileDebitAmount = @FileDebitAmount + @BatchDebitAmount
    end
    close @EntryDetail
30

    -- Third Batch: SWB cash
    set @EntryDetail = cursor for
    select
35      '6'
        + -- as RecordType,
        '27'
        + -- as TransactionCode,
        stb.CashDepositABA
40      + -- as ReceivingABA,
        cast(stb.CashDepositAccount as char(17))
        + -- as DFIAccountNumber,
        right('0000000000' + cast(round((sum(isnull(prt.PaymentAmount +

```

```

prt.AmountApplied,0)) * 100),0) as bigint) as varchar),10)
    -- as DollarAmount,
    as Part1,
    'askHarris.com      '
5      + -- as DestinationName
    '1'
    + -- as MediaDetermination,
    ''
    + -- as DiscretionaryDate,
10    '0'
    + -- as AddendaRecordIndicator,
    substring(@ChaseCompanyID, 1, 8)
    -- as OriginatingDFIID,
    as Part2,
15    cast(left(stb.CashDepositABA,8) as int) as EntryHash,
    isnull(sum(prt.PaymentAmount + prt.AmountApplied),0) as DebitAmount
from
    PaymentRecordTransaction prt
    join BillPay bp on bp.PaymentRecordID = prt.PaymentRecordid and bp.KioskPCID =
20    prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
    join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and bp.KioskPCID
    = bdt.KioskPCID and bp.DateStamp = bdt.DateTimeofBillScan
    join MainSecurity.dbo.KioskPCInfo kpc on prt.KioskPCID = kpc.KioskPCID
    join MainSecurity.dbo.KioskInfoMaster kim on kim.KioskInfoID = kpc.KioskID
25    join MainSecurity.dbo.StoreBank stb on stb.StoreID = kim.StoreID
where
    prt.PaymentTypeID = 1 and
    prt.QBTed = 10 and
    bdt.CompanyAccountTypeID = 3
30 group by
    stb.CashDepositABA, stb.CashDepositAccount

open @EntryDetail

35 fetch from @EntryDetail
into @OutputPart1, @OutputPart2, @DetailEntryHash, @DetailDebitAmount

if @@fetch_status = 0
begin
40    select @BatchCount = @BatchCount + 1
    -- Totals for Batch Control Record are figured in the loop
    -- PaymentType = 1 for checks
    -- CompanyAccountTypeID = 3 for SWB

```

```

-- record type 5
-- Batch Header
insert into ##Chase_ACH (OutputLine)
values
5      (      '5'
          + -- as RecordType,
          '200'
          + -- as ServiceClassCode,
          'askHarris.com '
10      + -- as CompanyName,
          ,
          + -- as CompanyDiscretionaryData,
          '4' + @ChaseCompanyID
          + -- as CompanyID,
15      'CCD'
          + -- as StandardEntryClassCode,
          'SWB CASH '
          + -- as CompanyEntryDescription,
          @FileXmitDate
20      + -- as CompanyDescriptiveDate,
          @OneDayDate
          + -- as EffectiveEntryDate,
          ,
          + -- as Settlement,
25      '1'
          + -- as OriginatorStatusCode,
          substring(@ChaseCompanyID, 1, 8)
          + -- as OriginatingDFIID,
          right('0000000' + cast(@BatchCount as varchar),7)
30      -- as BatchNumber
      )

```

```

-- record type 6
-- Entry Detail
35      select @BatchEntryCount = 0
          select @BatchEntryHash = 0
          select @BatchDebitAmount = 0

```

```

while @@fetch_status = 0
40      begin

```

```

          select @BatchEntryCount = @BatchEntryCount + 1
          select @BatchEntryHash = @BatchEntryHash + @DetailEntryHash

```

```

select @BatchDebitAmount = @BatchDebitAmount + @DetailDebitAmount

select @TraceNumber = @TraceNumber + 1
insert into ##Chase_ACH (OutputLine)
5 values (@OutputPart1 + right('0000000000000000' + cast(@TraceNumber as
varchar),15) + @OutputPart2 + right('0000000' + cast(@TraceNumber as varchar),7))
fetch next from @EntryDetail
into @OutputPart1, @OutputPart2, @DetailEntryHash, @DetailDebitAmount
end
10
-- record type 8
-- Batch Control
insert into ##Chase_ACH (OutputLine)
values
15 (
'8'
+ -- as RecordType,
'200'
+ -- as ServiceClassCode,
right('000000' + cast(@BatchEntryCount as varchar),6)
20 + -- as EntryCount,
right('00000000000' + cast(@BatchEntryHash as varchar),10)
+ -- as EntryHash,
right('0000000000000' + cast(cast(round((@BatchDebitAmount * 100),0) as
bigint) as varchar),12)
25 + -- as TotalDebitAmount,
'0000000000000'
+ -- as TotalCreditAmount,
'4' + @ChaseCompanyID
+ -- as CompanyID,
30 ,
+ -- as Reserved1,
,
+ -- as Reserved2,
substring(@ChaseCompanyID, 1, 8)
35 + -- as OriginatingDFI,
right('0000000' + cast(@BatchCount as varchar),7)
-- as BatchNumber
)

40 -- write batch total number to wire transfer table
insert into
MainSecurity.dbo.WireTransfers
(Payee, WireDate, WireAmount, Comment)

```

```

        values
            ('ACH: SWB Cash', dateadd(day,@OneDayOffset,getdate()),
            @BatchDebitAmount, 'Generated ' + convert(varchar,getdate(),101))

```

```

5      select @FileEntryCount = @FileEntryCount + @BatchEntryCount
      select @FileEntryHash = @FileEntryHash + @BatchEntryHash
      select @FileDebitAmount = @FileDebitAmount + @BatchDebitAmount
end

```

```

10     close @EntryDetail

```

```

-- Fourth Batch: non-SWB cash
set @EntryDetail = cursor for

```

```

15     select
        '6'
            + -- as RecordType,
        '27'
            + -- as TransactionCode,
20     stb.CashDepositABA
            + -- as ReceivingABA,
        cast(stb.CashDepositAccount as char(17))
            + -- as DFIAccountNumber,
        right('0000000000' + cast(cast(round((sum(prt.PaymentAmount + prt.AmountApplied)
25     * 100),0) as bigint) as varchar),10)
            -- as DollarAmount,
        as Part1,
        'askHarris.com'
            + -- as DestinationName
30     '1'
            + -- as MediaDetermination,
        ''
            + -- as DiscretionaryDate,
        '0'
            + -- as AddendaRecordIndicator,
35     substring(@ChaseCompanyID, 1, 8)
            -- as OriginatingDFIID,
        as Part2,
        cast(left(stb.CashDepositABA,8) as int) as EntryHash,
40     isnull(sum(prt.PaymentAmount + prt.AmountApplied),0) as DebitAmount
from
    PaymentRecordTransaction prt
join BillPay bp on bp.PaymentRecordID = prt.PaymentRecordid and bp.KioskPCID =

```







```

        + -- as ServiceClassCode,
        right('000000' + cast(@BatchEntryCount as varchar),6)
        + -- as EntryCount,
        right('0000000000' + cast(@BatchEntryHash as varchar),10)
5         + -- as EntryHash,
        right('000000000000' + cast(cast(round((@BatchDebitAmount * 100),0) as
        bigint) as varchar),12)
        + -- as TotalDebitAmount,
        '000000000000'
10         + -- as TotalCreditAmount,
        '5' + @ChaseCompanyID
        + -- as CompanyID,
        ,
        + -- as Reserved1,
15         ,
        + -- as Reserved2,
        substring(@ChaseCompanyID, 1, 8)
        + -- as OriginatingDFI,
        right('0000000' + cast(@BatchCount as varchar),7)
20         -- as BatchNumber
    )

    -- write batch total number to wire transfer table
    insert into
25         MainSecurity.dbo.WireTransfers
        (Payee, WireDate, WireAmount, Comment)
    values
        ('ACH: Utility Cash', dateadd(day,@TwoDayOffset,getdate()),
        @BatchDebitAmount, 'Generated ' + convert(varchar,getdate(),101))
30
        select @FileEntryCount = @FileEntryCount + @BatchEntryCount
        select @FileEntryHash = @FileEntryHash + @BatchEntryHash
        select @FileDebitAmount = @FileDebitAmount + @BatchDebitAmount
    end
35
    close @EntryDetail

    -- Fifth Batch: Service Charge Payouts
    set @EntryDetail = cursor for
40    select
        '6'
        + -- as RecordType,
        '22'

```

```

        + -- as TransactionCode,
stb.StorePaymentABA
        + -- as ReceivingABA,
cast(stb.StorePaymentAccount as char(17))
5      + -- as DFIAccountNumber,
right('0000000000' + cast(cast(round((sum(case bdt.ServiceAmt
        when 1 then .75
        when .5 then .25
        when 0 then 0
10     end) * 100),0) as int) as varchar),10)
        -- as DollarAmount,
        as Part1,
        'Kroger' ,
        + -- as DestinationName
15     '1'
        + -- as MediaDetermination,
        ''
        + -- as DiscretionaryDate,
        '0'
20     + -- as AddendaRecordIndicator,
substring(@ChaseCompanyID, 1, 8)
        -- as OriginatingDFIID,
        as Part2,
cast(left(stb.StorePaymentABA,8) as int) as EntryHash,
25     isnull(sum(case bdt.ServiceAmt
        when 1 then .75
        when .5 then .25
        when 0 then 0
        end),0) as CreditAmount,
30     isnull(sum(.25),0) as askSvcChg
from
    PaymentRecordTransaction prt
    join BillPay bp on bp.PaymentRecordID = prt.PaymentRecordid and bp.KioskPCID =
prt.KioskPCID and bp.DateStamp = prt.DateTimeofPayment
35     join BillDetailTransaction bdt on bdt.BillDetailID = bp.BillDetailID and bp.KioskPCID
= bdt.KioskPCID and bp.DateStamp = bdt.DateTimeofBillScan
    join MainSecurity.dbo.KioskPCInfo kpc on prt.KioskPCID = kpc.KioskPCID
    join MainSecurity.dbo.KioskInfoMaster kim on kim.KioskInfoID = kpc.KioskID
    join MainSecurity.dbo.StoreBank stb on stb.StoreID = kim.StoreID
40 where
    prt.PaymentTypeID = 1 and
    prt.QBTed = 10
group by

```

Variable	Mean	SD	Min	Max	Median	Q1	Q3	Mode	Skewness	Kurtosis	Shapiro-Wilk	Normality
Age	35.2	12.5	18	65	32	28	38	35	0.15	2.1	0.98	Normal
Gender	1.2	0.4	1	2	1	1	2	1	0.05	0.1	0.95	Normal
Education	12.5	2.1	9	16	12	11	13	12	0.12	1.8	0.97	Normal
Income	1500	500	500	3000	1200	800	1800	1000	0.25	3.5	0.92	Normal
Marital Status	1.5	0.5	1	2	1	1	2	1	0.08	0.2	0.96	Normal
Occupation	2.5	1.2	1	4	2	1	3	2	0.18	2.5	0.94	Normal
Health Status	1.8	0.6	1	2	1	1	2	1	0.03	0.05	0.99	Normal
Stress Level	3.2	1.5	1	5	3	2	4	3	0.22	3.8	0.91	Normal
Life Satisfaction	4.5	1.2	3	6	4	4	5	4	0.10	2.0	0.97	Normal
Work-Life Balance	3.8	1.0	2	5	3	3	4	3	0.14	2.3	0.96	Normal
Family Support	4.2	1.1	3	5	4	4	5	4	0.09	1.9	0.98	Normal
Community Involvement	2.8	1.3	1	5	2	1	3	2	0.20	3.6	0.93	Normal
Personal Growth	3.5	1.4	2	5	3	2	4	3	0.16	2.7	0.95	Normal
Overall Well-being	4.0	1.1	3	5	4	4	5	4	0.07	1.7	0.99	Normal

5

15

25

35

40

30

20

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

```

-- as BatchNumber
)

-- record type 6
5  -- Entry Detail
   select @BatchEntryCount = 0
   select @BatchEntryHash = 0
   select @BatchCreditAmount = 0
   select @TotalAHSvcChg = 0
10
   while @@fetch_status = 0
   begin

       select @BatchEntryCount = @BatchEntryCount + 1
15       select @BatchEntryHash = @BatchEntryHash + @DetailEntryHash
       select @BatchCreditAmount = @BatchCreditAmount + @DetailCreditAmount
       select @TotalAHSvcChg = @TotalAHSvcChg + @DetailAHSvcChg

       select @TraceNumber = @TraceNumber + 1
20       insert into ##Chase_ACH (OutputLine)
       values (@OutputPart1 + right('000000000000000' + cast(@TraceNumber as
varchar),15) + @OutputPart2 + right('0000000' + cast(@TraceNumber as varchar),7))
       fetch next from @EntryDetail
       into @OutputPart1, @OutputPart2, @DetailEntryHash, @DetailCreditAmount,
25 @DetailAHSvcChg
   end

   -- Write record to pay out askHarris service charge
30 select @TraceNumber = @TraceNumber + 1
   select @BatchEntryCount = @BatchEntryCount + 1
   select @BatchEntryHash = @BatchEntryHash + left(@AskHarrisSvcChgABA, 8)
   select @BatchCreditAmount = @BatchCreditAmount + @TotalAHSvcChg
   insert into ##Chase_ACH (OutputLine)
35 values
   (      '6'
         + -- as RecordType,
         '22'
         + -- as TransactionCode,
40     @AskHarrisSvcChgABA
         + -- as ReceivingABA,
         @AskHarrisSvcChgAccount
         + -- as DFIAccountNumber,

```

```

        right('0000000000' + cast(cast(round((@TotalAHSvcChg * 100),0) as int) as
varchar),10)
        + right('0000000000000000' + cast(@TraceNumber as varchar),15)
          + -- as DollarAmount,
5      'askHarris.com      '
          + -- as DestinationName
        '1'
          + -- as MediaDetermination,
        ''
10      + -- as DiscretionaryDate,
        '0'
          + -- as AddendaRecordIndicator,
        substring(@ChaseCompanyID, 1, 8)
          -- as OriginatingDFIID
15      + right('0000000' + cast(@TraceNumber as varchar),7)
    )

    -- record type 8
    -- Batch Control
20    insert into ##Chase_ACH (OutputLine)
    values
    (      '8'
          + -- as RecordType,
        '200'
25      + -- as ServiceClassCode,
        right('000000' + cast(@BatchEntryCount as varchar),6)
          + -- as EntryCount,
        right('00000000000' + cast(@BatchEntryHash as varchar),10)
          + -- as EntryHash,
30      '00000000000000'
          + -- as TotalDebitAmount,
        right('0000000000000' + cast(cast(round((@BatchCreditAmount * 100),0) as
bigint) as varchar),12)
          + -- as TotalCreditAmount,
35      '6' + @ChaseCompanyID
          + -- as CompanyID,
        '      '
          + -- as Reserved1,
        '      '
40      + -- as Reserved2,
        substring(@ChaseCompanyID, 1, 8)
          + -- as OriginatingDFI,
        right('0000000' + cast(@BatchCount as varchar),7)

```

```

-- as BatchNumber
)

-- write batch total number to wire transfer table
5 insert into
    MainSecurity.dbo.WireTransfers
    (Payee, WireDate, WireAmount, Comment)
    values
    ('ACH: Service Charges', dateadd(day,@TwoDayOffset,getdate()),
10 @BatchCreditAmount, 'Generated ' + convert(varchar,getdate(),101))

-- write askHarris number to wire transfer table
insert into
    MainSecurity.dbo.WireTransfers
15 (Payee, WireDate, WireAmount, Comment)
    values
    ('ACH: askHarris Fee', dateadd(day,@TwoDayOffset,getdate()),
    @TotalAHSvcChg, 'Generated ' + convert(varchar,getdate(),101))

20 select @FileEntryCount = @FileEntryCount + @BatchEntryCount
    select @FileEntryHash = @FileEntryHash + @BatchEntryHash
    select @FileCreditAmount = @FileCreditAmount + @BatchCreditAmount
end

25 close @EntryDetail

-- Sixth Batch: Cash Vouchers
set @EntryDetail = cursor for
select
30 '6'
    + -- as RecordType,
    '22'
    + -- as TransactionCode,
    stb.StorePaymentABA
35 + -- as ReceivingABA,
    cast(stb.StorePaymentAccount as char(17))
    + -- as DFIAccountNumber,
    right('0000000000' + cast(cast(round((sum(cvd.CashVoucherAmount) * 100),0) as
bigint) as varchar),10)
40 + -- as DollarAmount,
    as Part1,
    'Kroger '
    + -- as DestinationName

```

```

        '1'
            + -- as MediaDetermination,
        ''
            + -- as DiscretionaryDate,
5      '0'
            + -- as AddendaRecordIndicator,
        substring(@ChaseCompanyID, 1, 8)
            -- as OriginatingDFIID,
            as Part2,
10     cast(left(stb.StorePaymentABA,8) as int),
        isnull(sum(cvd.CashVoucherAmount),0) as VoucherTotal
from
    CashVoucherIssueDetailTransaction cvd
    join MainSecurity.dbo.KioskPCInfo kpc on cvd.KioskPCID = kpc.KioskPCID
15    join MainSecurity.dbo.KioskInfoMaster kim on kim.KioskInfoID = kpc.KioskID
    join MainSecurity.dbo.StoreBank stb on stb.StoreID = kim.StoreID
where
    cvd.QBTed = 10
group by
20    stb.StorePaymentABA,
    stb.StorePaymentAccount

open @EntryDetail

25  fetch from @EntryDetail
    into @OutputPart1, @OutputPart2, @DetailEntryHash, @DetailCreditAmount

    if @@fetch_status = 0
    begin
30        select @BatchCount = @BatchCount + 1
            -- Totals for Batch Control Record are figured in the loop
            -- record type 5
            -- Batch Header
        insert into ##Chase_ACH (OutputLine)
35        values
            (
                '5'
                    + -- as RecordType,
                '200'
                    + -- as ServiceClassCode,
40        'askHarris.com '
                    + -- as CompanyName,
                ,
                + -- as CompanyDiscretionaryData,

```

```

        '7' + @ChaseCompanyID
            + -- as CompanyID,
        'CCD'
            + -- as StandardEntryClassCode,
5      'CA VOUCHER'
            + -- as CompanyEntryDescription,
        @FileXmitDate
            + -- as CompanyDescriptiveDate,
        @TwoDayDate
10      + -- as EffectiveEntryDate,
        , ,
            + -- as Settlement,
        '1'
            + -- as OriginatorStatusCode,
15      substring(@ChaseCompanyID, 1, 8)
            + -- as OriginatingDFIID,
        right('0000000' + cast(@BatchCount as varchar),7)
            -- as BatchNumber
    )

-- record type 6
-- Entry Detail
select @BatchEntryCount = 0
select @BatchEntryHash = 0
25  select @BatchCreditAmount = 0

while @@fetch_status = 0
begin

30      select @BatchEntryCount = @BatchEntryCount + 1
        select @BatchEntryHash = @BatchEntryHash + @DetailEntryHash
        select @BatchCreditAmount = @BatchCreditAmount + @DetailCreditAmount

        select @TraceNumber = @TraceNumber + 1
35      insert into ##Chase_ACH (OutputLine)
        values (@OutputPart1 + right('0000000000000000' + cast(@TraceNumber as
varchar),15) + @OutputPart2 + right('0000000' + cast(@TraceNumber as varchar),7))
        fetch next from @EntryDetail
        into @OutputPart1, @OutputPart2, @DetailEntryHash, @DetailCreditAmount
40      end

-- record type 8

```



```

-- Batch Control
insert into ##Chase_ACH (OutputLine)
values
(      '8'
5      + -- as RecordType,
      '200'
      + -- as ServiceClassCode,
      right('000000' + cast(@BatchEntryCount as varchar),6)
      + -- as EntryCount,
10     right('0000000000' + cast(@BatchEntryHash as varchar),10)
      + -- as EntryHash,
      '000000000000'
      + -- as TotalDebitAmount,
      right('000000000000' + cast(cast(round((@BatchCreditAmount * 100),0) as
15 bigint) as varchar),12)
      + -- as TotalCreditAmount,
      '7' + @ChaseCompanyID
      + -- as CompanyID,
      ,
20     + -- as Reserved1,
      ,
      + -- as Reserved2,
      substring(@ChaseCompanyID, 1, 8)
      + -- as OriginatingDFI,
25     right('0000000' + cast(@BatchCount as varchar),7)
      -- as BatchNumber
    )

-- write batch total number to wire transfer table
30 insert into
      MainSecurity.dbo.WireTransfers
      (Payee, WireDate, WireAmount, Comment)
values
      ('ACH:  Cash  Voucher',  dateadd(day,@TwoDayOffset,getdate()),
35     @BatchCreditAmount, 'Generated ' + convert(varchar,getdate(),101))

      select @FileEntryCount = @FileEntryCount + @BatchEntryCount
      select @FileEntryHash = @FileEntryHash + @BatchEntryHash
      select @FileCreditAmount = @FileCreditAmount + @BatchCreditAmount
40 end

close @EntryDetail

```

-- Seventh Batch: Southwestern Bell CTX

-- cursor for RMR (remittance) EDI info

declare @EDIinfo varchar(80)

5 declare EDI cursor for

select

'RMR\*IV\*'

+

cast(max(bdt.BillAccountNumber) as varchar)

10 + '\*\*' +

cast((sum(prt.PaymentAmount) - max(bdt.ServiceAmt)) as varchar)

+ '\' as EDIinfo

from

PaymentRecordTransaction prt

15 join BillPay bp on prt.KioskPCID = bp.KioskPCID and prt.PaymentRecordID =

bp.PaymentRecordID and prt.DateTimeofPayment = bp.DateStamp

join BillDetailTransaction bdt on bdt.KioskPCID = bp.KioskPCID and bdt.BillDetailID

= bp.BillDetailID and bdt.DateTimeofBillScan = bp.DateStamp

where

20 bdt.companyaccounttypeid = 3 and

bdt.QBTed = 10

-- and

-- prt.PaymentAmount > 0

group by bdt.billdetailid, bdt.kioskpcid

25 having (sum(prt.PaymentAmount) - max(bdt.ServiceAmt)) > 0

order by EDIinfo

open EDI

fetch from EDI into @EDIinfo

30

if @@fetch\_status = 0

begin

select @BatchCount = @BatchCount + 1

35

-- record type 7

-- Addenda

-- containing EDI 820 data for SWB payments

-- generate batch credit amount for EDI and CTX detail

40 select @BatchCreditAmount = sum(dummy)

from

(select

isnull((sum(prt.PaymentAmount) - max(bdt.ServiceAmt)),0) as dummy

```

from
    PaymentRecordTransaction prt
    join BillPay bp on prt.KioskPCID = bp.KioskPCID and prt.PaymentRecordID
= bp.PaymentRecordID and prt.DateTimeofPayment = bp.DateStamp
5    join BillDetailTransaction bdt on bdt.KioskPCID = bp.KioskPCID and
    bdt.BillDetailID = bp.BillDetailID and bdt.DateTimeofBillScan = bp.DateStamp
    where
        bdt.CompanyAccountTypeID = 3 and
        bdt.QBTed = 10
10    group by bdt.billdetailid, bdt.kioskpcid
    having (sum(prt.PaymentAmount) - max(bdt.ServiceAmt)) > 0) foo

-- print EDI header and address info
15    select @OutputLine = 'ISA*00*      *00*      *03*798520847 *01*006968523
    *' + @FileXmitDate + '*' + @FileXmitTime + '*U*00401*' + @EDIControlNbr + '*0*P*>\'
    while len(@OutputLine) > 80
    begin
        insert into ##SWB_EDI (OutputLine)
20        values (left(@OutputLine, 80))

        select @OutputLine = substring(@OutputLine, 81, 200)
    end

25    select @OutputLine = @OutputLine + 'GS*RA*798520847*006968523*20' +
    @FileXmitDate + '*' + @FileXmitTime + '*01*X*004010\' + 'ST*820*0001\'
    while len(@OutputLine) > 80
    begin
30        insert into ##SWB_EDI (OutputLine)
        values (left(@OutputLine, 80))

        select @OutputLine = substring(@OutputLine, 81, 200)
    end

35    select @OutputLine = @OutputLine + 'BPR*C*' + cast(@BatchCreditAmount as
    varchar) +
    '*C*ACH*CTX*01*111000609*DA*30802536134*1'+substring(@ChaseCompanyID, 1,
    9)+'**01*021000021*DA*ICP323076769*20' + @OneDayDate + '*VEN\'
40    while len(@OutputLine) > 80
    begin
        insert into ##SWB_EDI (OutputLine)
        values (left(@OutputLine, 80))

```

```

        select @OutputLine = substring(@OutputLine, 81, 200)
    end

    select @OutputLine = @OutputLine +
5  'TRN*01*000000000000001*1'+substring(@ChaseCompanyID, 1, 9)+'\' +
    'N1*PE*SOUTHWESTERN BELL TELEPHONE*01*006968523\'+'N3*P.O. BOX 930170\'
    while len(@OutputLine) > 80
    begin
        insert into ##SWB_ED1 (OutputLine)
10      values (left(@OutputLine, 80))

        select @OutputLine = substring(@OutputLine, 81, 200)
    end

15      select @OutputLine=@OutputLine+'N4*DALLAS*TX*753930170\'+'N1*PR*ASK
    HARRIS*03*798520847\' + 'N3*2700 POST OAK SUITE 600\' +
    'N4*HOUSTON*TX*77056\'+'ENT*1\'

    while len(@OutputLine) > 80
20      begin
        insert into ##SWB_ED1 (OutputLine)
        values (left(@OutputLine, 80))

        select @OutputLine = substring(@OutputLine, 81, 200)
25      end

    declare @EDICount int
    select @EDICount = 10

30      -- loop over EDI cursor and fill temp table
    while @@fetch_status = 0
    begin
        select @EDICount = @EDICount + 1

35      select @OutputLine = @OutputLine + @EDInfo
        while len(@OutputLine) > 80
        begin
            insert into ##SWB_ED1 (OutputLine)
40          values (left(@OutputLine, 80))

            select @OutputLine = substring(@OutputLine, 81, 200)
        end
    end

```

```

-- don't forget to fetch the next one
fetch next from EDI into @EDIinfo
end

5      select @OutputLine = @OutputLine + 'SE*' + cast(@EDICount+1 as varchar) + '*0001\'
      + 'GE*1*01\' + 'IEA*1*000000001\'
      while len(@OutputLine) > 80
      begin
10          select @EDICount = @EDICount + 1

          insert into ##SWB_EDI (OutputLine)
          values (left(@OutputLine, 80))

          select @OutputLine = substring(@OutputLine, 81, 200)
15      end

          select @EDICount = @EDICount + 1

          insert into ##SWB_EDI (OutputLine)
          values (@OutputLine)
20

      -- record type 5
      -- Batch Header
      insert into ##Chase_ACH (OutputLine)
25      values
      (
          '5'
          + -- as RecordType,
          '200'
          + -- as ServiceClassCode,
          'askHarris.com '
30          + -- as CompanyName,
          ,
          + -- as CompanyDiscretionaryData,
          '1' + @ChaseCompanyID
          + -- as CompanyID,
35          'CTX'
          + -- as StandardEntryClassCode,
          'SWB PAYMT '
          + -- as CompanyEntryDescription,
40          @FileXmitDate
          + -- as CompanyDescriptiveDate,
          @OneDayDate
          + -- as EffectiveEntryDate,

```

```

        ' '
        + -- as Settlement,
        '1'
        + -- as OriginatorStatusCode,
5      left(@ChaseCompanyID, 8)
        + -- as OriginatingDFIID,
        right('0000000' + cast(@BatchCount as varchar),7)
        -- as BatchNumber
    )
10
    -- record type 6
    -- Entry Detail
    -- CTX entry detail is not like the others

15    declare @AddendaCount int
    select
        @AddendaCount = count(*)
    from
        ##SWB_EDI

20    select @BatchEntryCount = 1
    select @BatchEntryHash = 2100002

    select @TraceNumber = @TraceNumber + 1
25    insert into ##Chase_ACH (OutputLine)
    values
    (
        '6'
        + -- as RecordType,
30    '22'
        + -- as TransactionCode,
        '021000021'
        + -- as ReceivingABA,
        -- remember, BatchEntryHash is set above.
35    cast('ICP323076769' as char(17))
        + -- as DFIAccountNumber,
        right('0000000000' + cast(cast(round(@BatchCreditAmount * 100,0) as bigint)
as varchar),10)
        -- as DollarAmount,
40    + right('0000000000000000' + cast(@TraceNumber as varchar),15) +
        right('0000' + cast(@AddendaCount as varchar),4)
        + -- number of addenda records
        cast('Southwestern Bell' as char(16))

```

```

        + -- as DestinationName
    ' '
        + -- as Reserved
    ' '
5      + -- as DiscretionaryData,
    '1'
        + -- as AddendaRecordIndicator,
    left(@ChaseCompanyID, 8)
10     -- as OriginatingDFIID,
    + right('0000000' + cast(@TraceNumber as varchar),7)
)

-- cursor for wrapped EDI info
15 declare EDI2 cursor for
select
    OutputLine
from
    ##SWB_EDID
20 order by
    id

open EDI2
fetch from EDI2 into @EDIinfo
25 select @EDICount = 0

-- print addenda records
while @@fetch_status = 0
begin
30     select @EDICount = @EDICount + 1
        select @TraceNumber = @TraceNumber + 1

        insert into ##Chase_ACH (OutputLine)
        values
35     (
        '7'
            + -- as RecordType,
        '05'
            + -- as AddendaType,
        @EDIinfo
40         + -- as Info,
        right('0000' + cast(@EDICount as varchar),4)
            + -- as AddendaSeqNo,
        right('0000000' + cast(@TraceNumber as varchar),7)
    )

```

```

-- as TraceNumber
)

-- don't forget to fetch the next one
5      fetch next from EDI2 into @EDIinfo
      end
      select @EDICount = @EDICount + 1

close EDI2
10    deallocate EDI2

-- record type 8
-- Batch Control
insert into ##Chase_ACH (OutputLine)
15    values
      (      '8'
            + -- as RecordType,
            '200'
            + -- as ServiceClassCode,
20      right('000000' + cast(@EDICount as varchar),6)
            + -- as EntryCount,
            right('00000000000' + cast(@BatchEntryHash as varchar),10)
            + -- as EntryHash,
            '00000000000000'
25      + -- as TotalDebitAmount,
            right('0000000000000' + cast(cast(round(((@BatchCreditAmount * 100),0) as
bigint) as varchar),12)
            + -- as TotalCreditAmount,
            '1' + @ChaseCompanyID
30      + -- as CompanyID,
            ,
            + -- as Reserved1,
            ,
            + -- as Reserved2,
35      left(@ChaseCompanyID, 8)
            + -- as OriginatingDFI,
            right('0000000' + cast(@BatchCount as varchar),7)
            -- as BatchNumber
      )

40
-- write batch total number to wire transfer table
insert into
      MainSecurity.dbo.WireTransfers

```



```

                (Payee, WireDate, WireAmount, Comment)
            values
                ('ACH: SWB Payment', dateadd(day,@OneDayOffset,getdate()),
@BatchCreditAmount, 'Generated ' + convert(varchar,getdate(),101))
5
                select @FileEntryCount = @FileEntryCount + @EDICount
                select @FileEntryHash = @FileEntryHash + @BatchEntryHash
                select @FileCreditAmount = @FileCreditAmount + @BatchCreditAmount

10 end

close EDI
deallocate EDI

15 -- record type 9
-- File Control
insert into ##Chase_ACH (OutputLine)
values
(
    '9'
20
        + -- as RecordType,
        right('000000' + cast(@BatchCount as varchar),6)
        + -- as BatchCount,
        right('000000' + cast(((@FileEntryCount + 2 + (@BatchCount * 2)) / 10) + 1 as
varchar),6)
25
        + -- as BlockCount,
        right('00000000' + cast(@FileEntryCount as varchar),8)
        + -- as EntryCount,
        right('00000000' + cast(@FileEntryHash as varchar),10)
        + -- as EntryHash,
30
        right('000000000000' + cast(cast(round((@FileDebitAmount * 100),0) as bigint) as
varchar),12)
        + -- as TotalDebitDollar,
        right('000000000000' + cast(cast(round((@FileCreditAmount * 100),0) as bigint) as
varchar),12)
35
        + -- as TotalCreditDollar,
        ,
        -- as Reserved
    )

40 select @TraceNumber = 10 - (@FileEntryCount + 2 + (@BatchCount * 2))%10
if @TraceNumber < 10
    while @Tracenumber > 0
    begin

```



The present invention contemplates a variety of steps which may be all inclusive or utilized on a selective basis to implement the method. The machine and the method may be designed so that the operator initially selects the language which will be utilized on instructions appearing on the video screen 3 and in the audio companion instructions also appearing on the video screen 3. The operator simply touches the touch screen monitor 3 at the appropriate location to designate a language, such as English, Spanish, French, or the like. Thereafter, the operator is asked if the payment is to be for one bill or a number of bills. If the customer elects to process only one bill for payment, the customer chooses by application through the touch screen monitor 3 at the appropriate location the "BILL PAY" identification. Alternatively, if the customer would like to pay multiple bills, an indicator on the touch screen monitor 3, such as "COMPLEX" or "MULTI-BILLS" will appear on the screen and the operator applies a finger or otherwise touches the appropriate location on the screen 3 to reflect his/her selection.

The machine and method of the present invention are preferably designed to charge the operator a nominal fee for the electronic payment process. Accordingly, the customer is asked to understand and agree to the terms of the operation of the machine and the practice of the method, including agreement to the payment of a service charge by touching the "AGREE" field on the screen 3. Alternatively, the customer may touch "CANCEL" to return to the "MAIN MENU" reflected on the screen.

The next sequence of steps in the operation of the machine and the practice of the method is for the operator to select a utility or other company, sometimes referred to as "billor", to whom payment is to be made. Accordingly, the identification of the utility or other company will appear on the screen 3 and the operator will touch the identification of such company at the

appropriate location on the screen 3.

After the billor has been selected by the operator, the bill is scanned through the bill scanner 6 after an instruction to do so is reflected on the screen 3 concurrently and preferably with a similar audio instruction.

5           The machine and method of the present invention contemplate providing a discount of the service charge for senior citizens, disabled or other preselected groups of individuals or entities under an obligation to satisfy payment of bills by the billors (such people and the group sometimes referred to as "payors"). For example, if a discount is to be given to a senior citizen, a query regarding same appears on the screen 3 and confirmed through audio companion signal  
10   to the operator for an inquiry to confirm "senior citizen" status. The operator then touches the appropriate location on the screen to confirm "YES" or "NO". Verification of senior citizen status is established by insertion by the operator of a driver's license or other personal identification in the card reader portion of the key pad on the optical check reader 5. Thereafter, the computer in the machine calculates the amount due from data extracted from the bill inserted  
15   into the machine, and the operator is asked on the screen 3 how payment will be made. For example, "CASH", "CASH/CHECK", "CHECK", options may appear on the screen 3 and the operator may touch the appropriate location for manner of payment.

If cash payment is selected, the total balance is displayed on the screen 3 and the appropriate amount of cash is deposited by the operator into the bill acceptor 6. The bill  
20   acceptor 6 will extract data therefrom and the computer will calculate and confirm complete, exact payment, or, if over payment, will generate an instruction to the bill acceptor 6 to generate a cash voucher, as required. Alternatively, the machine will indicate on the touch screen 3 an

excess of money which may be applied as either a credit against future bills of the billor or payment of another bill by the billor or another billor in subsequent operation of the machine. Alternatively, if change is due, a "cash" voucher may be issued by query to the operator on the video screen 3 and confirmation by touching the appropriate location on the screen. A credit  
5 voucher will then be generated by the machine. Alternatively, over-payment of the bill may be either applied directly to the bill by touching the appropriate location in response to a query appearing on the screen 3 or, alternatively, confirming by touch to the screen 3 application of the over-payment to the next utility bill to be paid on the machine.

10 If the check option is desired by the operator for payment of all or part of the bill, the query appears on the screen 3 and confirmation thereof is indicated by the operator touching the appropriate location on the screen. The check is then scanned as previously described through introduction of the check into the reader 5.

Another feature of the present invention is the ability to scan and process the check used to pay a bill without the necessity of the person or entity owing the bill, or the operator of the  
15 machine being required to actually sign the check and/or fill in the amount of the check. The check is scanned and processed as an ACH transaction. If the operator desires to pay the bill by use of a check, the computer generates a graphics and video instruction to the operator that it is not necessary to fill out the check or sign it. The customer then is instructed to introduce the check through the scanner and it is returned to the customer. An image of the check is  
20 recorded in the data base in the computer and picture of the customer is taken. The customer is then asked for authorization to use the driver's license or other identification document of the customer for identification purposes and the driver's license is then scanned for information

thereon. The customer is then asked to confirm through touch on the touch screen in an appropriate location to confirm that the customer is an authorized signor on the checking account. A manual confirmation by the customer of the amount to be paid through the check may be made by introducing the monetary amount of the check by application of fingers to the

5 key pad. The ACH transaction is then processed in batch or real time through an electronic banking ACH process for deduction of the amount from the customer's checking account as indicated for the transaction. If a check is used for payment regardless of whether or not it is signed and/or filled out, proper identification of the operator is prompted by the operator confirming permission to use a driver's license by response to a query appearing on the screen

10 3. The operator touches the "OK TO USE DRIVERS LICENSE" or other appropriate wording, by touch to corresponding field on the screen 3. The drivers license or other document of the operator is then scanned and information is stored in the computer thereon. The computer will then read a magnetic strip or the like on the back of the drivers license or other identification card. Confirmation of authorization to sign on the checking account reflected on the check is

15 also processed through the machine. The customer is asked to so confirm by appropriate message appearing on the touch screen 3 and the customer is prompted to apply touch in the appropriate location to a message reading "CONFIRM" or the like.

The full amount of the total bill payment is displayed on the screen 3 after processing through the computer. If the customer wishes to pay the full amount of the bill through check,

20 the customer will press the field "PAY FULL" appearing on the screen 3.

The machine and method will accommodate partial payment of the full amount displayed by check upon so indicating by the customer in the appropriate field on the screen 3.

The checking account information extracted by the machine, including the amount paid on the bill is transmitted in real time or batch concurrently from the financial institution issuing the check and holding the account of the operator to the account of the payor at the billor. The check then is returned through the optical check reader 5 to the customer and a receipt is generated by the computer and printed on paper through the printer 8 in the machine. Alternatively, a cash voucher may be issued through the printer 8 or the amount can be treated as a "credit" for use by the operator in paying another bill. If another bill is desired to be paid, the process is repeated. Now with reference to the logic flow chart of Figs. 2A and 2B, the human operator or customer initiates bill payment by pressing the start button 1 on the face of the kiosk machine 100 and is asked via a message on the video screen blank and/or via audio message if the customer agrees to a service charge of a specified amount, for example \$1, at message 2A. Pressing of the "no" (or "agree") button 2B, (or failure to press the "yes" button 2C) within a specified short time, say, 5 seconds, will cause movement to the exit field 3 and shut down of the service function to the operator. An affirmation of the service charge 2A through activation of the "yes" button 2C will open to the operator the selection of the utility company 4 either by pressing in an appropriate place on the screen corresponding to an identification of the selected utility, by pressing a selection button, or the like. The machine next tests if the electronic bill scanner is properly working in field 5. If the scanner is not working properly an "out of order" message 6 is generated and an alert message is sent via e-mail or other signal and an "out of order" is displayed on the video screen in function 7.

If the scanner check step 5 indicates proper functioning 8, the actual bill payment functions are initiated at function 9. As shown in Fig. 2A, the payment step may be that of

actually paying a bill of a utility or making a deposit for security or as a pre-payment against a future bill. The operator is prompted at step 9 to select either the "deposit" or the "bill" payment option by touching an appropriate place on the video screen or by pressing a button or otherwise. If the deposit sub-function 9B is selected, the machine then determines if the selected utility company allows pledges or donations at function 10. If this answer is affirmative 10A, the operator is asked if he/she desires to make such donation at 10B. If the operator responds affirmatively at 10C, the operator then enters on the key pad or by touching the video screen the amount of the donation 11. However, if the operator elects not to make a donation, the "no" button 10D is pressed, or the equivalent position on the video screen is contacted by the operator to lead to the next operation, discussed below.

If no donation amount is entered by the customer at function 10D, the operator is prompted via audio/video message to confirm the amount of the donation through step 12.

After the amount is confirmed at 13, the customer is asked if he/she is a senior citizen 14. This question is also asked as the next step in the event that the utility or other billor does not accept pledges or donations at 10 or if the operator does not wish to make a donation 10D.

If the operator answers affirmatively that he/she is a senior citizen 14A, the operator is asked to apply his/her driver's license or other identification document to an optical or similar scanner 15. If the results of the scanning of the driver's license at 15 indicates that the age of the operator is over a specified amount, such as 55 years at step 16, by affirmative answer "yes" 16A, a deduction of a specified amount of the service charge for example, 50¢ is activated at 17A. If the age requirement for the service charge deduction is not satisfied at 16B, i.e., the operator is not a senior citizen 14B, the computer program will cause the machine to



automatically add a service charge in a specified amount, for example, \$1.00 at step 18. Alternatively, if the customer is a senior citizen and the driver's license scan confirms that the customer is over a specified age, the deduction at step 17A is made after application of the service charge at 18.

5           The next grouping of steps in the operation of the present bill payment method and machine involves the collection and transfer of funds for payment of the selected bill of the utility or other company. The operator is asked through step 19 whether payment will be made through cash 19A or check 19B. If the operator presses the appropriate button on the key pad or touches the appropriate field in the video screen for payment by check 19B, the check is  
10   scanned at 20 through the appropriate scanner. The data base in the computer is then checked at field 21 to see if the check has been previously utilized or processed for payment at another location, such as through direct processing or the like. The name on the check reflected through the check scan at 20 is compared against the name on the driver's license scanned at 15 to verify the name on the checking account reflected on the check at step and field 22. The driver's  
15   license is again scanned at 15A, as in 15 and a photograph of the operator is digitally taken through a digitized photo reproduction means also at 15A. Confirmation that the operator is an authorized person on the account is made at 24, such as by communication and confirmation through the bank or other financial institution (not shown).

          The operator then is asked at 25 if either full or partial payment of the bill is to be made.  
20   If full payment is to be made by indication of "yes" at 26, the signal from the check scan function 20 is retained at 27 and a receipt is printed and received by the operator at 28. However, if the customer indicates only partial payment of the bill at 29, the amount of the

check is entered at 30 and confirmed at 31. This step is repeated through repeat function 32 until the check amount 30 is confirmed.

Returning to the determination of whether payment is to be made by cash or check, at function 19, if the operator indicates through the key pad or touches the appropriate point on the video screen, that the payment is to be made through cash, 19A, the cash is inserted through the cash receiving device at 23. The operator then confirms full cash payment by pressing "done" on the key board or by touching such position on the video screen. The device then confirms either correct payment, over payment, or under payment at function 34. If the bill has been under paid at 34, the operator is asked to either insert additional money at 33 or may pay the difference at 35 through check, in which case, the check is scanned at 20 and the steps are followed as though complete payment had been made through check. If correct payment has been made at 36, a receipt for the amount is made through step 28. Likewise, if the operator elects not to pay the remaining amount by check at step 35, through 37, the final receipt is printed at 28 for the amount of the payment, even though it is less than the full amount owed for the bill to be paid.

If the operator has overpaid the bill through an indication at field 34, the operator is prompted to select a cash voucher for future usage or to apply the amount of the overpayment to the next bill at step 38. If a "cash voucher" is indicated, the cash voucher is printed at 39. Alternatively, the amount of the over payment is added to the total in the data base at 40. In either event of 39 or 40, a receipt is then printed at 28. Alternatively, an overpayment indicated at 34 may be applied to the next bill to be paid through the system at 36, in which event the customer indicates a desire to pay another bill at 37 and the bill payment cycle is continued at

41 by selecting the second utility company at 4 and repeating the steps described above..

Returning to the step of either depositing an amount or paying a bill at step 9, if the bill function is selected, the bill is scanned at 42 and checked for proper scanning through 43. If it has not been properly scanned, 44, any scanning errors are identified and corrected at 42, and the procedure continues through step 10 to determine allowance of pledges or donations. If the utility bill is properly scanned at 42, the account digits are checked and verified at 43. If the digits cannot be verified at 43, a "fail", 43, is indicated, and the process is repeated through steps 9 and 43. If the digits are checked and verified, 45, step 10 is initiated to determine if the utility company allows pledges and donations and the sub-procedures described above, are followed.

Turning, now, to Fig. 2B, the deposit step 9A is further detailed. The deposit is made at 9B and the bill account number is entered by the operator on the key pad or by application of touch to the proper field on the video screen at 9C. The account number is confirmed by the computer at 9D. If there is no confirmation, 9E, the deposit cycle is repeated, 9B. If the account number is confirmed, 9F, the check digits are confirmed 9G and, if negative, the deposit cycle 9B is repeated. If the check digits pass, the driver's license is scanned and a photo image on the driver's license is extracted at step 9H. The deposit amount then is entered at 9I and confirmed 9J. The deposit amount is re-entered if there is no confirmation of the amount. If the amount is confirmed, the sub-step is terminated at 9K.

Fig. 2B also depicts the steps and sub-steps for determining the errors 42 in properly scanning the bill through step 40. If errors 42 are determined, the computer checks for error count in steps 42A and 42B. If no errors are determined, the system will return to the scan bill step 40. If an error has been determined at 42A or 42B, the computer will indicate a "bad" read,

42C on the video terminal and the customer will be asked to enter the account number manually  
42D. The number is then reflected on the video screen with a "is this correct?" query, 42E, and  
if not, 42F, the number is again manually entered at 42D. If the correct number is established,  
42G, the check digits are confirmed at 42H and if not confirmed, the number is again manually  
5 re-entered at 42D. The manual amount, 42I, is confirmed at 42J until passed at 42K, at which  
time return is established to sub-step 40 and passage through check digits 42, 44.

### **MAGNETIC INK CHARACTER RECOGNITION COMPONENT**

Magnetic ink character recognition ("MICR") is used to scan checks which are tendered  
by the customer for payment of bills. MICR involves two fundamental steps in the recognition  
10 process. First, the magnetizable ink forming a recognizable character must be magnetized to  
create a magnetic image of the character. Secondly, the magnetic image must be sensed or read  
and identified as a character in accordance with pattern recognition techniques, well known to  
those skilled in the art. The character field is generally divided into a plurality of discrete,  
vertical lines or segments, each line or segment being individually magnetized. A common  
15 magnetization technique is to apply a sinusoidally-varying magnetic field over the character,  
where each sinusoidal cycle is intended to magnetize one discrete segment of the character.  
This technique has been shown by experience to facilitate the reading and recognition of the  
character.

When the check bearing in-coded characters of magnetizable ink is magnetized or  
20 written upon, it is transported past a write station having a write head that generates the  
sinusoidally varying magnetic field. The transportation of the check past the written station is  
generally by mechanical means, such as a read drum having an outer circumferential edge that

grips the document and rotationally transports it past both the right and read stations. It is important that the mechanical transport means be driven at a uniform, consistent velocity to correlate the position of the character field in the write station with the timed actuation of the write head for properly spaced magnetization to occur. Such systems are well known to those skilled in the art of electronic reading of characters on legal instruments, such as checks, and the like. The reader is directed to U.S. Patent No. 4,087,789 for a more detailed description of typical prior art apparatuses. Such apparatuses are commercially available through a number of corporations, such as Burroughs Corporation, of Detroit, Michigan.

In operation, the MICR units include the magnetizing head, a magnetic read head and circuitry for recognition of the characters. In operation, the check is passed over the magnetized head, which magnetizes the magnetic particles in the ink, and over the read head, which detects the magnetization of the ink, and over the read head, which detects the magnetization of the magnetized particles and transmits representative signals to the recognition circuitry. Frequently, a drive mechanism is provided to drive the check through a channel in the reader pass the magnetizing and read heads. In many commercial embodiments, the channel extends the length of the reader. One end of the check is inserted into one end of the channel and is driven the length of the reader along the channel until the opposite end of the check exits the opposite end of the channel, where it can be retrieved by the customer.

Optical character readers may also be used to read the information encoded on the checks. Such readers incorporate an optical character read head and circuitry for recognition of the optical characters. Optical characters typically conform to a predetermined specification such as ANSI-X-3.17.

The present invention contemplates usage by the customer of credit or bank cards in lieu of cash or checks. Such cards contain information encoded on a magnetic strip on one side of the card. The strips typically include up to 3 tracks. Information is then coded on the tracks in accordance with certain standards, such as ANSI-X-4.16 and ISO3554, specifications for magnetic strip inclosing. Track one has been developed for use by the air transportation industry, track two for the banking industry and track three for the thrift industry. Magnetic strip readers include a magnetic strip read head and circuitry for recognition of the encoded information. In operation, the magnetic strip of the credit or bank card is passed over the read head.

Magnetic strip readers are used in retail establishments for processing purchases made with a credit or bank card. Typically, these devices are located close to a cash register and include a slot or channel, open at both ends, through which a sales clerk at the point of sale slides or "swipes" the card. Many commercial devices include a MICR reader and a magnetic strip reader combined as a single unit. However, the electronic components of the MICR reader must be sufficiently far away from the magnetic strip reader to avoid erasing the magnetic strip data on a card passing through the magnetic strip slot. The area around both ends of the device must be kept relatively free of other objects to provide unrestricted access to the slots or channels, particularly for checks, which comprise of paper medium which are easily bent or torn. Accordingly, when a combined magnetic strip reader and a MICR reader are embodied in the machine and method of the present invention, it should be compact for accommodation into a comparatively small area. Typical of the commercially available combined readers is the SCAN TEAM™® 8300 of Raco Industries.

Although the invention has been described in terms of specified embodiments which are set forth in detail, it should be understood that this is by illustration only that the invention is not necessarily limited thereto, since alternative embodiments and operating techniques will become apparent to those skilled in the art in view of the disclosure. Accordingly, 5 modifications are contemplated which can be made without departing from the spirit of the described invention.

Approved for release